MASTER EN SCIENCES INFORMATIQUES

Comparaison qualitative et quantitative de deux bases de données relationnelles

Delos, Geneviève

Award date:
1990

Awarding institution:
Universite de Namur

Facultés
Universitaires
N. D. de la Paix
NAMUR

Institut d'informatique

Comparaison qualitative et quantitative de deux bases de données relationnelles

par Geneviève DELOS

Promoteur
Professeur J. RAMAEKERS

Mémoire présenté en vue de
l'obtention du grade de
Licencié et Maître en
Informatique

Année académique 1989 - 1990

Avant de vous présenter le travail qui clôture les cinq années amenant à la Licence et Maîtrise en Informatique, je tiens à remercier toutes les personnes qui m'ont permis de le réaliser.

Ma gratitude s'adresse en premier lieu à Monsieur Jean Ramaekers, le promoteur de ce mémoire, ainsi qu'à tous les autres professeurs qui m'ont fourni matière à intense réflexion durant ces cinq années d'étude.

Mes remerciements vont ensuite à Monsieur Jean-Michel Adam et à Madame Brigitte Nizet qui m'ont accueillie chez Unisys (Evere) où j'ai passé plus de cinq mois de stage très enrichissant.

J'exprime aussi ma reconnaissance à toute l'équipe Mini-Micro, et plus particulièrement au groupe UNIX de chez Unisys qui m'a permis de travailler dans les meilleures conditions possibles. Parmi cette sympathique équipe, Monsieur Eric Lamy a accepté de me suivre et de me conseiller durant toute la durée de ce mémoire.

Enfin, que tous ceux et celles qui m'ont aidée et entourée tout au long de la rédaction de ce travail trouvent également ici l'expression de ma reconnaissance.

Résumé — Abstract

De nos jours, l'utilisation de gestionnaires de bases de données ne pose aucun problème. Cependant, pouvoir choisir le "bon" gestionnaire de base de données est plus problématique. Ce mémoire tente d'éclaircir quelque peu ce problème en réalisant une comparaison qualitative et quantitative de deux produits relationnels : ORACLE et INFORMIX. Durant un stage de près de cinq mois, nous avons pu étudier ces gestionnaires de bases de données et ainsi réaliser une comparaison théorique pour servir la partie qualitative, et ensuite créer une base de données afin d'établir des tests de performances qui constituent la comparaison quantitative; ces deux dernières phases étant introduites par une partie théorique relative à la conception de bases de données et à la réalisation de test de mesures de performances.

Nowadays, the use of databases management systems don't give any problems. But, to choose and use the "good" one, is more problematic. That's why this work try to give some ideas to solve this problem with the realization of a qualitative and quantitative comparison between two relational produtcs : ORACLE and INFORMIX. A training period of 5 months give us the possibility to study this two relational databases management systems, to make a theorical analysis for the qualitative aspect and also to realize a database and performing tests for the qualitative comparison; this second part of the comparison is introduced by a theorical presentation of databases's conception and benchmarks.

Sommaire

Liste des figures

1. Introduction

Actuellement, la plupart des entreprises sont informatisées et leurs données traitées par d'importants gestionnaires de bases de données. Afin de toujours être à la pointe ou devenir encore meilleures, ces entreprises recherchent le système qui leur convient le mieux. C'est un peu dans cette perspective que ce travail a été réalisé.

Le but de ce mémoire est donc une comparaison qualitative et quantitative des deux bases de données relationnelles que sont ORACLE d'ORACLE Corporation et INFORMIX développé par INFORMIX Software INC. L'aspect qualitatif sera servi par une comparaison théorique, et l'aspect quantitatif par une série de mesures de performances.

Les deux produits étudiés lors d'un stage de près de cinq mois chez Unisys (Evere), ainsi que l'environnement dans lequel s'est déroulé ce travail sera présenté au chapitre 2. Dans celui-ci, nous allons donc définir les versions et les éléments des deux bases de données relationnelles sur lesquels l'étude portera, ainsi que la machine sur laquelle se sont déroulés les tests de performances pour l'aspect quantitatif.

L'aspect qualitatif de cette comparaison sera présenté au chapitre 3 par la comparaison théorique. Pour réaliser celle-ci, nous avons procédé à une analyse systématique d'un ensemble pertinent de critères qui constituent une grille d'analyse. Nous verrons ainsi successivement toutes les possibilités des deux systèmes au niveau : d'informations générales (sur la représentation des données, le contrôle d'intégrité, l'interrogation, la modification des données), de l'accès pour les utilisateurs (c'est-à-dire le fonctionnement interactif, le fonctionnement en langage hôte, les générateurs d'écrans et de rapports), de la protection (qui comprend la protection des accès, le chiffrement de données, les accès multiples, le recouvrement et les sauvegardes de données) et d'éléments permettant la mise au point pour de meilleures performances (c'est-à-dire la taille d'une base de données, des données dynamiques et statistiques d'utilisation de la base de données, le fonctionnement de l'optimiseur, la réorganisation).

Afin de procéder aux mesures de performances, nous avons créé une base de données. Mais pour réaliser ces deux phases, un chapitre

théorique (le chapitre 4) nous montre les fondements de la conception d'une base de données (c'est-à-dire l'analyse conceptuelle, la conception logique et la conception physique). Ce chapitre explique également en quoi peuvent consister des mesures de performances. En effet, nous y trouverons la définition d'un "benchmark", quand et comment le réaliser.

L'élaboration de la base de données testée, expliquée au chapitre 5 suit la théorie développée au chapitre précédent. Dans ce chapitre, sont également présentées la mise en oeuvre et la création de la base de données sur la machine de développement.

Au chapitre 6, nous arrivons à la partie concernant la comparaison quantitative. Avant d'analyser les résultats sous différentes formes (pour ORACLE, pour INFORMIX, pour ORACLE et INFORMIX) et leurs conclusions. nous présentons le principe général de ces mesures ainsi que diverses hypothèses de travail. Tous les résultats seront analysés théoriquement et en fonction de la base de données testée. Enfin, dans ce chapitre, un dernier point sera abordé, il concerne divers autres résultats observés durant les mesures et des perspectives d'approfondissement possibles de tous ces travaux.

Nous arrivons alors au chapitre 7 qui donne une conclusion finale pour cette comparaison.

2. Présentation du contexte d'étude

2.1 Introduction

Ce chapitre va présenter les versions et les éléments des deux logiciels ORACLE et INFORMIX qui ont été étudiés. On précisera également l'environnement technique dans lequel s'est déroulé ce travail.

2.2 Présentation générale des bases de données relationnelles étudiées ainsi que les langages les supportant

Les deux **gestionnaires de bases de données relationnelles** (ou **RDBMS** pour **Relational Data Base Management System**) qui seront comparés tout au long de cette étude sont donc d'une part celui développé par INFORMIX SOFTWARE INC. et qui se nomme **INFORMIX** et d'autre part, celui proposé par ORACLE CORPORATION et que l'on appelle **ORACLE**. Ces deux RDBMS comprennent un langage de développement afin de créer et manipuler les données. Dans ce but, INFORMIX SOFTWARE INC. a créé un langage de 4^{ième} génération : INFORMIX-4GL. ORACLE CORPORATION quant à lui a développé un langage **SQL (System Query Language)** qui se divise en plusieurs parties; chaque partie du langage traitant d'un problème particulier (par exemple : **SQL*PLUS** pour la simple manipulation des données, **SQL*FORMS** pour la génération d'écrans, **SQL*NET** pour la gestion des réseaux, ...). Il est important de noter cependant que ces deux langages de développement ont une même racine qui est la norme **ANSI** pour SQL [BIJI,sd].

Avant d'entamer plus avant cette comparaison, il ne serait peut-être pas inutile d'amener quelques précisions sur ce qu'est un langage de 4^{ième} génération (ou 4GL), et, peut-on affirmer que le langage SQL développé par ORACLE CORPORATION est un **langage de 4^{ième} génération** ? Il existe actuellement sur le marché beaucoup d'implémentations du concept

de **4GL**. Cependant, celles-ci nous paraissent relativement incomplètes car elles ne reprennent en général qu'une seule facette de ce type de langage. Dès lors, pour définir un tel langage, on peut s'attacher à différents points de vue (ceux-ci étant aussi importants les uns que les autres) :

1. Ce langage doit être convivial c'est-à-dire qu'une requête écrite dans ce langage s'apparente fortement au langage parlé;

2. Ce langage doit travailler par transactions, donner la possibilité de créer ses propres transactions. Une **transaction** pouvant être définie comme une suite de plusieurs instructions ayant les propriétés suivantes : atomicité, cohérence (si la base de données est cohérente avant la transaction, elle l'est encore après), isolation (il n'y a aucune connexion entre les transactions), et durabilité (quand une transaction est terminée. ses effets sont durables) [HAINA,90];

3. Ce langage doit apporter une meilleure productivité qu'un langage de 3^{ième} génération (comme COBOL par exemple); c'est-à-dire que pour obtenir un résultat semblable à celui d'un langage de 3^{ième} génération, beaucoup moins d'instructions sont nécessaires (et on peut même aller jusqu'à dix fois moins d'instructions avec un temps d'exécution raisonnable) [DONKI,sd];

4. Ce langage doit être flexible, c'est-à-dire qu'il peut être utilisé à des fins diverses telles que la création d'écrans, de rapports [DONKI,sd], ...

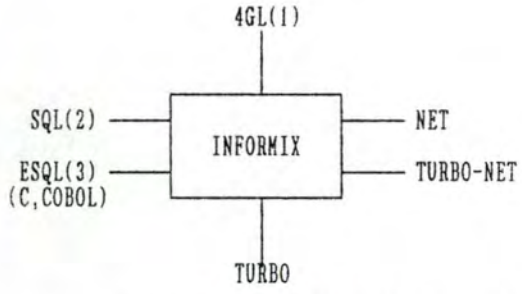
Au vu de toutes les caractéristiques qui viennent d'être énoncées, on peut dire que le langage SQL développé par ORACLE CORPORATION, pris dans son ensemble est un langage de 4^{ième} génération.

2.3 Les divers composants des RDBMS étudiés

2.3.1 D'un point de vue commercial

Ces deux gestionnaires de bases de données relationnelles vont être analysés sur un type de machine bien particulier : le U 6000/50 (dont les caractéristiques sont données au point 2.4). Voici dès lors aux figures 1

et 2 les divers éléments composant ces deux RDBMS et qui sont commercialement disponibles sur ce type de machine [UNIS2,89], [UNIS3,89].

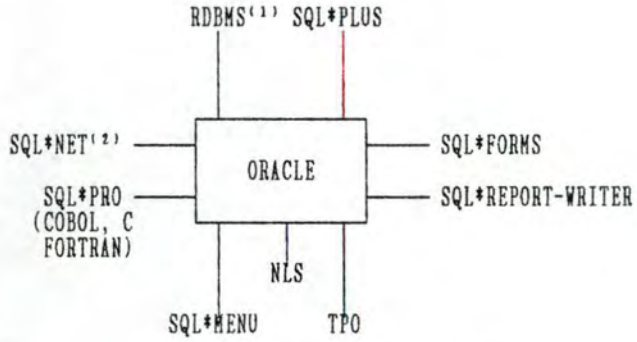


où : ESQL = Embedded SQL language
(1) Le 4GL est composé de :
- 4GL
- 4GL Run-Time
- 4GL Rapid Development System
- 4GL RDS Run-Time
- 4GL Interractive Debugger
- 4GL RDS ID

(2) Le SQL est composé de :
- SQL RDBMS
- SQL Run-Time

(3) Le ESQL est composé de :
- ESQL/C
- ESQL/C Run-Time
- ESQL/COBOL pour compilateur RM Cobol 85
- ESQL/COBOL pour compilateur MF Cobol 85

Figure 1 : Les composants d'INFORMIX



où NLS = National Language Support
TPO = Transaction Processing Option
(1) Le RDBMS comprend :
- SQL*REPORT
- SQL*LOADER
- SQL*DBA

(2) Le SQL*NET comprend :
- SQL*NET avec les protocoles TCP/IP
- SQL*NET avec les protocoles Async
- SQL*NET avec les protocoles DECnet

Figure 2 : Les composants d'ORACLE

2.3.2 D'un point de vue technique

Ces langages se perfectionnent rapidement, et de plus, possèdent de multiples facettes. Dès lors, dans le cadre de cette comparaison, il a fallu s'attacher à une version particulière de ces gestionnaires de bases de données relationnelles ainsi qu'à certains de leurs composants.

Seront donc étudiés :

Pour INFORMIX :

- INFORMIX-4GL, version 1.10.03K
- Load Utility INFORMIX SQL, version 2.10

Pour ORACLE version 5:

- SQL*PLUS, version 2.0
- SQL*FORMS, version 2.0
- ORACLE Data Loader, version 5.1.22

A partir de ces éléments précisés ci-dessus, on peut se définir un schéma technique (figure 3) qui reprend les différentes facettes des RDBMS qui seront étudiées :

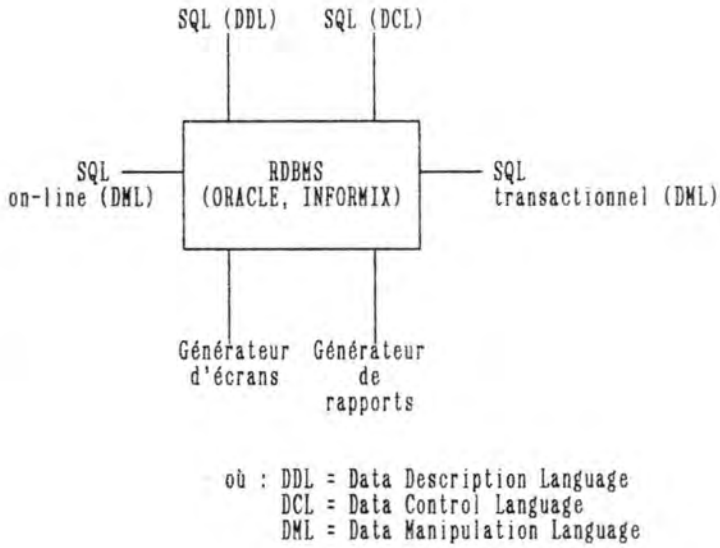


Figure 3 : Schéma technique des RDBMS

La question de savoir pourquoi se limiter précisément à ces éléments là pourrait être posée. La réponse à y apporter est relativement simple. En effet, les éléments qui nous paraissent les plus intéressants à

étudier sont la gestion d'une base de données, la génération d'écrans et de rapports ainsi que la gestion d'une base de données installée en réseau. A l'heure actuelle, il semble que se soient ces quatre points qui déterminent le choix en matière de système de gestion de bases de données. Le seul cas non étudié dans le cadre de ce travail est celui de la gestion en réseau car ce domaine mérite à lui seul de faire l'objet d'une étude bien particulière.

A propos d'INFORMIX, une précision supplémentaire doit être apportée. En effet, le 4GL-RDS, le 4GL-ID et le Turbo auraient pu être étudiés, mais pour une question de disponibilité, c'est sur le 4GL que va se porter l'analyse.

2.4 La machine de développement : le U 6000/50

2.4.1 Caractéristiques générales

L'analyse a donc été réalisée sur un **U 6000/50** qui fait partie de la série U 6000 développée par Unisys. Cet ordinateur à simple processeur est basé sur un micro-processeur intel 80386. Selon les besoins, cette machine peut s'intégrer dans un réseau (conforme aux standards OSI, travaillant avec des équipements IBM dans un environnement **SNA**, ou encore avec des systèmes **DEC VAX** dans un réseau **DEC NET**). Les deux environnements d'exploitation très répandus que sont **MS-DOS** et **UNIX** sont également supportés par cette machine [UNIS1,89], [UNIS4,89].

2.4.2 Caractéristiques spécifiques

Le système d'exploitation du U 6000/50 sur lequel l'analyse a été réalisée est un **UNIX** version V.3.2. Cette machine a été configurée de telle manière qu'elle possède une mémoire de 16 mégabytes (dont 8 Mb de mémoire cache et 8 Mb sont réservés pour les tampons) et supporte en moyenne 18 utilisateurs.

2.5 Conclusion

Nous avons donc vu dans ce chapitre que l'étude se réalisera sur une machine à simple processeur 80386 avec un système d'exploitation UNIX multi utilisateurs. D'autre part, après avoir précisé ce qu'était un langage de 4ième génération, nous avons expliqué quels seraient la version et les éléments des deux logiciels ORACLE et INFORMIX qui seront étudiés d'une manière théorique et/ou pratique.

3. Comparaison théorique

3.1 Introduction

Ce troisième chapitre va vous proposer une comparaison théorique, c'est-à-dire sur base des manuels disponibles, des deux RDBMS. Pour ce faire, une grille d'analyse a tout d'abord été élaborée. Cette dernière est totalement subjective car elle essaye de visualiser les éléments importants, utiles de chaque RDBMS.

3.2 Informations générales

3.2.1 La représentation des données

3.2.1.1 Valeur nulle et compression de données

La **valeur nulle** est la valeur "non défini" que l'on peut retrouver dans un schéma conceptuel lorsque l'on spécifie qu'un attribut ne doit pas obligatoirement recevoir une valeur. Pour un RDBMS, une valeur nulle indique qu'aucune valeur n'a été assignée à une colonne particulière dans une ligne particulière d'une table. Cette valeur se distingue d'un zéro, d'une valeur caractère n'ayant aucune signification pour l'utilisateur, et d'un ou plusieurs blancs [ISI1,87], [ISI2,87]. Idéalement, pour de meilleures performances, la représentation interne d'une valeur nulle devrait prendre aussi peu de place que possible, et en tout cas, pas toute la place définie pour l'attribut.

La **compression de données**, quant à elle, est un avantage appréciable pour toute base de données de taille importante. En effet, cela permet de réduire l'espace mémoire, disque utilisé par les données, et donc d'améliorer les performances du système.

Nous allons maintenant voir dans le tableau de la figure 4 ce qu'il en est pour INFORMIX et ORACLE en ce qui concerne ces deux caractéristiques [ORAC1,86], [ISI1,87] :

	INFORMIX	ORACLE
Valeur nulle		- Aucune place mémoire
Compression des données	- Suppression automatique des blancs précédants et suivants	- Suppression automatique des blancs précédants et suivants - Principe du clustering des tables - Compression de clé d'index

Figure 4 : Valeur nulle et compression de données

A propos de la compression des données pour ORACLE, quelques précisions peuvent être apportées:

1. Pour une table, la compression des données ne peut pas s'appliquer telle quelle; par contre, l'**administrateur de la base de données** (ou **DBA**, ou **Data Base Administrator**) peut donner aux tables une organisation physique particulière : le format **CLUSTERED** (qui sera détaillé lors de la comparaison quantitative). Pour ce format de table, ORACLE définit automatiquement une **clé d'index**, et l'administrateur peut demander que cette clé d'index soit comprimée ou non [ORAC1,86].

2. Lors de la création d'un index (qui n'est pas une opération réservée à l'administrateur de la base de données), on peut spécifier si on désire que ORACLE fasse une **compression de la clé** de cet index. Cette compression se réalise en deux phases selon l'algorithme suivant [ORAC1,86] :

a. The Forward Compression

ORACLE dans cette première phase identifie les sous ensembles de lettres qui sont similaires au début de deux valeurs de clé adjacentes. Il supprime ces lettres et en indique leur nombre.

b. The Rear Compression

Pour cette seconde phase, ORACLE retire à la fin de chaque clé les lettres qui sont identiques à celles de la clé précédente et suivante. Il garde le restant de la clé ainsi que sa longueur résultante.

Lorsque la clé est comprimée, ORACLE dispose donc de la clé actuelle (comprimée), de sa longueur résultant de la compression ainsi que du compteur obtenu lors de la première phase.

3.2.1.2 Représentation en longueur fixe ou variable

La représentation en **longueur variable** des données amène l'avantage du gain de place pour les données, mais entraîne une plus grande complexité de gestion pour le RDBMS qui pourrait pénaliser le temps d'accès. La représentation en **longueur fixe** des variables évite quant à elle cette difficulté de gestion, mais on perd dans ce cas l'avantage du gain de place. INFORMIX et ORACLE ont à ce sujet une philosophie totalement opposée puisque INFORMIX gère une représentation en longueur fixe, et ORACLE en longueur variable [ORAC1,86].

Idéalement, on pourrait dire que l'administrateur de la base de données devrait pouvoir choisir, selon ses besoins, entre ces deux représentations. Malheureusement, cette hypothèse n'est pas réaliste et pour les deux RDBMS étudiés le choix n'est pas autorisé.

3.2.1.3 Dimension maximale par attribut

Pour les deux gestionnaires de bases de données relationnelles étudiées, pour chaque type de données, une **dimension maximale** est définie. Celle-ci ne peut être modifiée. Toutefois, l'utilisateur a la possibilité de demander, pour le type de donnée associé à l'attribut qu'il définit, une dimension inférieure à la dimension maximale autorisée [ISI1,87], [ORAC4,86].

3.2.2 Contrôle de l'intégrité

Il s'agit ici des **règles d'intégrité** qui sont définies dans le **schéma conceptuel** et non de l'intégrité de la base de données en tant que reflet de la réalité. Ce deuxième point sera analysé ultérieurement lorsque seront étudiés tous les problèmes de sécurité et de recouvrement des données en cas d'incident sur ces dernières [BIJI,sd], [BODAR,88].

On se place au niveau application, et on peut distinguer trois types d'intégrité :

3.2.2.1 L'intégrité des données

C'est-à-dire avoir la possibilité de préciser quelles sont les valeurs que peuvent avoir un certain attribut; autrement dit, quelles sont les valeurs permises pour un champ d'un enregistrement de la base de données.

3.2.2.2 L'intégrité des entités

C'est-à-dire avoir le moyen d'indiquer, pour une table, le ou les champs (ou colonnes) dont la valeur des éléments doit être unique pour cette table.

3.2.2.3 L'intégrité référentielle

C'est-à-dire pouvoir préciser quels sont les champs (ou colonnes) d'une table qui correspondent à des champs d'une autre table.

Pour les deux gestionnaires de bases de données étudiés, ces trois types de contrôle d'intégrité sont réalisables. Cependant, la façon d'obtenir un résultat identique varie, et il est intéressant d'en voir les différences car un utilisateur peut, selon les cas, avoir une grande facilité ou difficulté pour obtenir ce résultat.

On obtient donc pour la réalisation de ces contrôles d'intégrité les possibilités suivantes (reprises à la figure 5) [ISI1,87], [ISI2,87], [ORAC4,86], [ORAC5,86] :

	INFORMIX	ORACLE
Intégrité des données	- Un champ ne peut avoir de valeur nulle - Instructions 4GL donnant un intervalle de valeur	- Un champ ne peut avoir de valeur nulle - Spécification de valeurs autorisées lors de la création d'un écran
Intégrité des entités	- Création d'un index unique - Type de donnée particulier pour ce champ : le SERIAL	- Création d'un index unique
Intégrité référentielle	- Instructions 4GL lors de la création d'un écran	- Apport d'une information lors de la création d'un écran

Figure 5 : Contrôle de l'intégrité

Nous remarquons qu'INFORMIX possède un type de données bien particulier : le **SERIAL**. Lorsque l'on travaille avec le **générateur d'écrans**, le champ qui possède un type de données SERIAL ne peut être modifié et la valeur de celui-ci s'incrémente automatiquement lorsque de nouveaux enregistrements sont créés. Ce type de données peut donc, dans ce cas, réaliser un contrôle automatique du 2^{ième} type de règle d'intégrité.

3.2.3 L'interrogation

Lorsqu'un utilisateur interroge la base de données, le RDBMS peut devoir parcourir un chemin à travers plusieurs tables pour lui fournir une réponse. Pour ce parcours, le RDBMS va donc faire des **liaisons** entre les différentes tables; ces dernières peuvent être multiples (2 ou plus), ou externes (on parle aussi de "**outer-joins**").

Aussi bien INFORMIX que ORACLE peuvent réaliser ces possibilités. La seule différence est que ORACLE ne permet la **liaison externe** que pour deux tables, INFORMIX quant à lui en admet deux ou plus quel que soit le type de liaison demandé [ISI1,87], [ISI3,87], [ORAC4,86], [UNIS1,88].

Un second élément important pour un utilisateur lors de l'interrogation, est la possibilité de pouvoir annuler la question posée. A ce moment, il est nécessaire de faire une distinction entre le niveau application et le niveau langage standard (c'est-à-dire les langages développés pour les deux gestionnaires de base de données). En effet, au niveau application, il est toujours possible, aussi bien avec INFORMIX que ORACLE, de créer une application qui demande à l'utilisateur s'il désire poursuivre l'interrogation, ou lui donner la possibilité d'annuler la question qu'il est en train de poser. Quant au niveau du langage standard d'INFORMIX, il est proposé sous forme de menus à l'utilisateur, et il est toujours possible d'annuler la question posée. Pour ORACLE, il faut distinguer le **SQL*PLUS** qui est en ligne et pour lequel une annulation est plus problématique, du **SQL*FORMS** qui est proposé également sous forme de menus à l'utilisateur et pour lequel l'annulation de la question posée n'engendre aucune difficulté.

3.2.4 La modification des données

Toute **modification** ou **suppression de données** devrait idéalement recevoir une confirmation avant toute exécution, afin d'éviter toute fausse manoeuvre qui pourrait avoir des conséquences importantes. Dans ce cas, il est également nécessaire de faire la distinction entre le niveau application, et le niveau langage standard. En résumé pour les deux langages nous pouvons observer ce tableau de la figure 6 :

	INFORMIX	ORACLE	
		SQL*PLUS	SQL*FORMS
Langage standard	- Toute destruction doit être confirmée - Après toute modification une manipulation est nécessaire afin de la confirmer	- Suivant la séquence d'instructions, une confirmation est nécessaire pour une modification ou suppression	- Aucune confirmation n'est demandée pour une destruction - Pour toute modification, une manipulation, donc une confirmation est nécessaire
Application	- On peut toujours construire une applica- où une confirmation est demandée pour toute modification ou suppression	- On peut toujours construire une applica- où une confirmation est demandée pour toute modification ou suppression	- Une confirmation est obligatoire pour toute modification ou suppression

Figure 6 : La modification des données

3.3 Accès pour les utilisateurs

3.3.1 Fonctionnement interactif

Aussi bien INFORMIX que ORACLE dispose d'un langage SQL interactif. Avec de telles possibilités, il est facile et très utile pour l'utilisateur de pouvoir disposer d'une **aide en ligne**. Nous pouvons examiner à la figure 7 les différences qui existent entre les deux langages :

	INFORMIX	ORACLE	
		SQL*PLUS	SQL*FORMS
Langage standard	- Menus disponibles - Aide en ligne	- Pas de menus - Aide en ligne	- Menus pour le Data Manipulation Language (DML) - Aide en ligne pour les touches clavier à utiliser
Application	- Possibilité de créer des menus - Possibilité de créer une aide en ligne		- Aide en ligne pour les touches clavier à utiliser et les actions à réaliser

Figure 7 : Fonctionnement interactif

Il faut ajouter à ce tableau que pour ORACLE, il existe également une possibilité de créer des menus pour les applications, et ce grâce au **SQL*MENU**.

3.3.2 Fonctionnement depuis un langage hôte

Un programmeur d'applications pourrait très bien vouloir écrire un logiciel en un autre langage que le SQL, 4GL de ORACLE ou INFORMIX et vouloir accéder aux bases de données gérées par ORACLE ou INFORMIX. Ces langages seront appelés langage hôte. Le nombre de langages hôtes différents utilisables est limité par la nécessité de disposer pour chacun d'un précompilateur pour permettre l'accès aux données. On dispose ainsi (voir figure 8) pour les deux RDBMS, pour la machine étudiée, des langages hôtes suivant [ISI3,87], [ORAC1,86] :

	INFORMIX	ORACLE
Langages hôtes	COBOL, C	COBOL, C, FORTRAN

Figure 8 : Les langages hôtes

Un élément supplémentaire doit être souligné; en effet, INFORMIX a une autre particularité : il permet, dans son propre code, de faire appel à des routines écrites en langage C. Connaissant les routines C, et le programme les appelant, INFORMIX fait une compilation et génère son code grâce à un éditeur de liens. Donc, on peut à la fois utiliser les informations d'une base de données gérée par INFORMIX à partir d'un langage hôte, et appeler des routines C à l'intérieur d'un programme écrit en INFORMIX-4GL.

3.3.3 Générateur d'écrans

Ce point est très important, car le générateur d'écran est l'outil qui va permettre une meilleure "convivialité" des applications pour les utilisateurs. Le résultat final doit être bon, mais pour le concepteur d'application, cela doit être idéalement le moins complexe possible. Sous ORACLE et INFORMIX, la philosophie de génération d'écrans est totalement différente.

3.3.3.1 Principe de génération d'écrans sous INFORMIX

Avec INFORMIX, il est possible de travailler avec des écrans définis par défaut, cette génération se faisant à partir d'une sélection dans un menu. Toute création d'écrans (non définis par défaut) ou toute modification d'écrans demande l'utilisation d'un **éditeur pleine page**. Ces écrans ainsi définis peuvent alors être intégrés dans une suite d'instructions; ce qui peut être très utile lors de la création d'applications. Lorsque l'on désire créer ou modifier un écran, une visualisation immédiate de celui-ci est possible; cela n'est pas à négliger si l'on désire obtenir un bon résultat. Bien entendu, pour une même application, le concepteur a la possibilité de placer sur un même écran des enregistrements provenant de plusieurs tables, mais aussi de placer sur plusieurs écrans des enregistrements d'une seule ou de plusieurs tables différentes [ISI1,87].

3.3.3.2 Principe de génération d'écrans sous ORACLE

Sous ORACLE, on peut également travailler avec des écrans définis par défaut, mais toute modification ultérieure de ces écrans, ou création de nouveaux écrans (non définis par défaut) se réalisent via des sélections dans divers menus. A l'opposé d'INFORMIX, les écrans ainsi définis ne peuvent pas être intégrés dans une suite d'instructions. Mais tout comme INFORMIX, ORACLE, sur un même écran, peut lier divers enregistrements provenant de plusieurs tables, ou encore, sur un même écran placer des enregistrements provenant d'une ou plusieurs tables [ORAC7,86], [ORAC8,86], [UNIS2,88].

On remarque donc que la grande différence réside en deux points : ORACLE travaille uniquement via des sélections dans des menus, ce qui semble plus simple que de travailler avec un éditeur pleine page comme INFORMIX, et deuxièmement, INFORMIX permet lui, d'intégrer des écrans définis dans une suite d'instructions. Cependant, malgré ces différences, les deux RDBMS donnent un résultat semblable quant à la conformité aux règles d'intégrité ainsi que cela a été analysé précédemment. Il faut enfin remarquer que si l'on désire utiliser le générateur d'écrans à partir d'un langage hôte, cela est impossible, aussi bien pour INFORMIX que pour ORACLE.

3.3.4 Générateur de rapports

Tout comme le générateur d'écrans, le générateur de rapports est un outil essentiel. INFORMIX et ORACLE disposent tout deux d'un outil utilisable à l'intérieur de la structure DBMS. Les rapports ainsi réalisés, peuvent contenir des textes libres. Cependant, une différence qui était constatée entre INFORMIX et ORACLE pour la génération d'écrans se retrouve ici : à savoir que pour INFORMIX, les instructions permettant de définir un rapport peuvent s'inclurent dans une suite d'instructions SQL. Mais, pour palier à cela, ORACLE fournit un second outil qui s'utilise à partir de la ligne de commande du système d'exploitation et qui permet d'inclure éventuellement des instructions DML [ORAC5,86], [ORAC9,86], [UNIS1,88], [ISI1,87].

3.4 La protection

3.4.1 La protection des accès

3.4.1.1 Usage de mots de passe

Pour pouvoir manipuler les données, ou accéder à certaines ressources machine, il est souvent demandé de fournir un nom d'utilisateur ainsi qu'un mot de passe lui correspondant. Le but de cette manipulation étant bien évidemment la protection de ces ressources ou données. Ainsi, le système d'exploitation UNIX s'enquiert de savoir si l'utilisateur qui désire travailler en a bien la permission; et pour cela, ce dernier doit lui donner un nom d'utilisateur et un mot de passe correct lui correspondant; après analyse de ces données par le système d'exploitation, l'utilisateur est autorisé ou non à accéder aux ressources.

Pour travailler avec les informations d'une base de données gérée par INFORMIX, aucune autre identification que celle exigée par le système d'exploitation n'est demandée. Par contre, ORACLE lui demande un nom d'utilisateur et un mot de passe lui correspondant pour autoriser l'accès aux informations de la base de données. La façon dont ce nom d'utilisateur et le mot de passe correspondant sont gérés ainsi que les privilèges qu'ils accordent seront discutés au point suivant [ORAC1,86].

3.4.1.2 Définition du profil

Dans ce paragraphe, nous allons examiner de quelle manière sont accordés les privilèges de manipulation de données ainsi que la possibilité ou non pour un utilisateur de transmettre à des tiers ceux qui lui ont été accordés. A ce niveau, les deux RDBMS ont de nombreux points communs. Tout deux, ils ont une **protection décentralisée** des données, c'est-à-dire que chaque utilisateur qui crée une table peut protéger les informations qu'elle contient face aux autres en définissant des **privilèges**. Pour certaines manipulations, ces privilèges peuvent même s'accorder au niveau des champs des enregistrements d'une table, et donc pas seulement au niveau de la table. De plus, lorsqu'un utilisateur a créé une table et qu'il accorde des privilèges à un tiers sur cette table, il peut également donner à ce tiers la possibilité de transmettre les privilèges qu'il a reçu.

En ce qui concerne les possibilités d'actions de l'administrateur de la base de données, celui-ci a accès à beaucoup plus d'informations qu'un autre utilisateur, et notamment à des informations permettant la gestion des données. Ainsi, il a accès à certaines tables qui montrent l'ensemble des privilèges accordés à l'ensemble des utilisateurs de la bases de données ainsi que les éléments sur lesquels sont accordés ces privilèges. Ces tables particulières font partie d'un ensemble de données que l'on appelle le **dictionnaire des données** (ou **DD**, ou **Data Dictionnary**). L'administrateur a également la possibilité de transmettre tous ses privilèges à un ou plusieurs autres utilisateurs, mais ces derniers ne pourront jamais les transmettre à leur tour.

A ce niveau de définition du profil, il existe cependant une grande différence entre INFORMIX et ORACLE. En effet, comme cela a été montré au paragraphe précédant, ORACLE protège l'ensemble de ses données par un nom d'utilisateur et un mot de passe associé. C'est l'administrateur de la base de données qui donne la possibilité à un futur utilisateur de manipuler les informations de la base de données; c'est donc lui qui distribue les noms d'utilisateur. Suivant l'identification propre à ORACLE, un utilisateur possède certains privilèges qui lui ont été accordés par l'administrateur et éventuellement par d'autres utilisateurs [ORAC1,86].

3.4.2 Le chiffrement des données

A l'heure actuelle, le problème de la sécurité devient de plus en plus important. Dès lors, la possibilité de pouvoir disposer d'un outil de chiffrement automatique des données serait un avantage appréciable. Malheureusement, ni INFORMIX, ni ORACLE n'offre une telle possibilité.

3.4.3 Accès multiples

3.4.3.1 Nombre d'utilisateurs

Ainsi que cela a été précisé plus haut, les systèmes sont analysés sur une machine U 6000/50 dont le système d'exploitation est UNIX qui admet par définition le multitâches. Dès lors, plusieurs utilisateurs pourraient en même temps vouloir travailler avec les informations de la base de données et même avoir besoin de manipuler les mêmes données au même moment. Le nombre d'utilisateurs simultanés autorisé théoriquement par le système d'exploitation est de 32; mais ainsi que nous l'avons vu précédemment, la machine utilisée est paramétrée dans ce cas pour autoriser 18 utilisateurs.

INFORMIX pour sa part n'impose aucune limitation quant au nombre d'utilisateurs simultanés. ORACLE non plus, cependant, il donne la possibilité à l'administrateur de fournir en paramètre d'initialisation, le nombre maximum d'utilisateurs en mémoire cache. Par défaut, ce nombre est de 30, mais aucune limite maximale n'existe. Comme on peut le deviner, ce paramètre peut être important pour une mise au point, et fournir un temps de réponse minimum pour toute requête demandée par les utilisateurs [ORAC1,86].

Pour terminer ce paragraphe, nous pouvons signaler que les deux systèmes de bases de données, dans les versions étudiées n'autorisent pas l'accès simultané à plusieurs bases de données différentes.

3.4.3.2 Verrouillage des données et traitement des interblocages

a. Verrouillage des données

Etant donné que plusieurs utilisateurs peuvent travailler en même temps sur la base de données et qu'ils peuvent vouloir accéder en même temps à certaines informations, il est nécessaire de prévoir un système de verrouillage des données afin d'éviter, par exemple, que deux utilisateurs,

au même moment, ne modifient une même donnée. Ainsi, pour INFORMIX et ORACLE, il existe un **blocage automatique des données** au niveau des enregistrements d'une table et au niveau des tables toutes entières. Le choix entre le verrouillage d'un enregistrement d'une table ou celui de la table est réalisé par le RDBMS, et cela en fonction de la requête SQL demandée. Mais en plus de cela, il est possible de demander explicitement aux deux RDBMS de bloquer une table. Par rapport à INFORMIX et ORACLE, on peut définir trois types de **verrouillage explicite** :

1. Le mode partagé

Un et un seul utilisateur peut réaliser des insertions, mise à jour ou suppression. Les autres peuvent cependant accéder aux informations de la table, mais uniquement en lecture. Ce mode de protection va empêcher tout blocage en mode exclusif (qui est expliqué dans le point suivant) de la table, mais par contre, il autorise tout verrouillage en mode partagé de cette même table.

2. Le mode exclusif

Un et un seul utilisateur a accès à la table. Ce mode exclusif va empêcher tout autre blocage de cette table.

3. Le mode mise à jour partagée

Il s'agit ici d'un mode partagé; il ne concerne pas une table tout entière, mais permet le verrouillage de plusieurs enregistrements d'une table en même temps. Ce mode de blocage des données n'autorise pas le verrouillage en mode exclusif de la ressource. Il s'agit donc d'un mode partagé au niveau des enregistrements et non plus de la table toute entière. Ce mode est principalement choisi lorsque l'on fait des mises à jour.

INFORMIX et ORACLE ne disposent pas tout deux de ces trois possibilités, en effet pour ces deux RDBMS, à la figure 9, on peut remarquer que l'on dispose de [ISI2,87], [ORAC1,86]:

b. Traitement des interblocages

Le **traitement des interblocages** ou **deadlocks** est également un point crucial dans ces systèmes de gestion de bases de données. Ainsi, par exemple, deux processus pourraient avoir bloqué des ressources ou données et attendre mutuellement que l'autre débloque les siennes. Ce problème peut être vu sous deux angles différents. En effet, le système peut soit prévenir

les interblocages et donc faire en sorte que ceux-ci ne se produisent pas, ce qui peut provoquer une diminution des performances car les solutions ne sont peut-être pas appliquées au bon moment; soit les détecter et les solutionner automatiquement, ce qui n'est pas simple non plus.

	INFORMIX	ORACLE
Mode partagé	oui	oui
Mode exclusif	oui	oui
Mode "Share-Update"	non	oui

Figure 9 : Verrouillage des données

ORACLE détecte les interblocages et les solutionne automatiquement. Lorsqu'il se rend compte qu'il existe une situation de deadlocks, le dernier processus à avoir demandé la ressource est "renvoyé en arrière", c'est-à-dire que sa demande d'acquisition des ressources est refusée et il doit recommencer sa transaction. Un autre processus peut dès lors acquérir la ressource.

En ce qui concerne INFORMIX, malheureusement, nous ne disposons pas au moment précis de cette rédaction de documents permettant d'établir sa stratégie en situation d'interblocage.

3.4.4 Recouvrement et sauvegarde des données

Le problème du **recouvrement des données**, ou **recovery** est également un point important dans la gestion d'un système de bases de données. En effet, à tout moment, l'intégrité de la base de données doit être respectée; c'est-à-dire que celle-ci doit refléter un état exact du système réel et qu'elle doit être capable de restituer toutes les informations qu'on y a enregistrées et elles seulement. Or, personne n'est à l'abri d'un crash disque ou encore d'une rupture de l'alimentation électrique. Pour qu'un recouvrement des données puissent se réaliser après qu'un incident ait eu lieu, il faut que le système de gestion de la base de données aie tous les renseignements nécessaires et donc, avoir organisé correctement son **back-up**.

Il existe divers types d'**incidents** : des incidents locaux à des transactions (déconnexion accidentelle d'un terminal par exemple), des incidents du système comme la coupure de courant, ou encore des incidents affectant le support de la base de données (disque, ...); et tous ces types d'incidents ne sont pas solutionnés de la même manière.

3.4.4.1 Recouvrement des données suite à un incident local à une transaction ou du système

Afin de permettre le recouvrement des données, on peut avoir créé un "**audit trail**". Un "audit trail" est une table qui ne contient pas de données, mais des renseignements sur ce que font les utilisateurs. INFORMIX et ORACLE permettent la création de telles tables qui gardent une trace de toutes les modifications apportées aux données, mais elle doit être explicitement demandée. La raison principale à cela est que, étant donné le travail supplémentaire que cela implique pour le RDBMS, cela va engendrer, on peut le deviner, un ralentissement des performances du système. L'audit trail se crée par table et peut être demandé aussi bien par l'administrateur de la base de données que par n'importe quel utilisateur. En cas d'incident, l'audit trail va donc pouvoir être utilisé pour reconstruire une table ayant été altérée, et ce sur base d'une sauvegarde réalisée au moment ou l'audit trail a été demandé [ORAC3,86].

ORACLE, en plus de tout cela dispose d'une procédure automatique de détection de terminaisons anormales des processus : il s'agit de la procédure de "**Clean-Up**". Celle-ci balaye périodiquement la mémoire afin de détecter toutes les terminaisons anormales de processus. Lorsqu'un tel événement est détecté, cette procédure nettoie la mémoire mais conserve tous les paramètres dans l'état où ils étaient lors de l'incident [ORAC1,86], [UNIS3,88].

3.4.4.2 Recouvrement des données suite à un incident affectant le support

Le principe de recouvrement des données dans ce cas doit être différencié entre INFORMIX et ORACLE. En effet, nous ne disposons malheureusement que très peu d'informations pour INFORMIX. Pour ce dernier, il existe une instruction SQL qui, grâce aux informations présentes sur un back-up, va rétablir l'intégrité de la base de données, mais uniquement pour les opérations qui auraient été confirmées. De plus, si à un moment donné un utilisateur désire vérifier qu'il n'existe aucune

incohérence entre les tables auxquelles il a accès et leur index, il en a la possibilité, et peut même corriger automatiquement les incohérences qui auraient été découvertes [ISI2,87], [ISI3,87].

Pour ORACLE, nous allons entrer un peu plus dans les détails. Ce RDBMS construit, si on le lui demande des fichiers **AIJ (After Image Journal)**. Ces fichiers contiennent une copie de tous les blocs modifiés qui sont écrits dans la base de données. Après installation par l'administrateur de tous les paramètres nécessaires, ces fichiers peuvent être utilisés par une procédure spéciale, la procédure AIJ, qui va mettre à jour la copie de back-up de la base de données qui reflète celle-ci quand le processus de journalling a commencé. Une fois cette copie mise à jour, la base de données retrouve son intégrité, et tout utilisateur peut la manipuler dans un état cohérent. Le processus de mise à jour d'un back-up par la procédure qui utilise les fichiers AIJ se réalise en deux phases [ORAC1,86], [UNIS3,88].

a. Première phase en général

Construction d'une liste des transactions qui n'ont jamais été confirmées afin de les éliminer du journal.

b. Deuxième phase en général

Modification de la base de données en faisant appel à tous les blocs qui ne comprennent pas de transactions incomplètes trouvées en première phase.

c. Première phase en détails

- La procédure AIJ contrôle tous les paramètres qui lui ont été fournis.
- Elle ouvre le premier fichier de la base de données
- Elle affiche le nom des extensions de la base de données
- Elle recherche dans chaque fichier toutes les transactions qui n'ont jamais été confirmées.
- Elle fait apparaître un message quand elle a terminé.
- L'utilisateur acquièsce ce dernier.

- Elle affiche alors un résumé des données trouvées dans le journal. Ce résumé comprend une liste des transactions qui n'ont jamais été terminées; chacune d'elle est affichée en une simple ligne constituée de l'identificateur de la transaction, ainsi que du numéro de bloc vu par la transaction.

d. Deuxième phase en détail

- Si il n'y a pas de transactions terminées
  Alors STOP

- Si l'option "NO-APPLY" est spécifiée, cela signifie que l'on ne désire pas remettre à jour la base de données
  Alors STOP

- La procédure AIJ ouvre toutes les extensions de la base de données.

- Le journal est à nouveau lu en entier en utilisant les noms de fichiers donnés en première phase.

- La base de données est mise à jour avec les blocs du journal qui contiennent les transactions confirmées.

- Quand la procédure AIJ commence a examiner un fichier, le nom de celui-ci est affiché.

- Quand elle a terminé d'examiner un fichier, elle affiche le nombre de blocs (dont la taille est donnée par un paramètre propre à ORACLE) qui sont réellement utilisés dans le fichier.

- Quand tous les fichiers du journal ont été examinés, le numéro de séquence journal est mis à jour dans la base de données.

- La procédure AIJ ferme les extensions de la base de données.
- Elle affiche un message de fin.

Suite à un incident, un utilisateur pourrait très bien vouloir retrouver la base de données dans un état précédant toutes les modifications qu'il y avait apportées. Dans ce but, ORACLE permet la création de fichier "**Before Image**"; et grâce à une instruction SQL, on peut

retrouver la base de données dans un état cohérent, mais ne contenant plus les modifications.

3.5 Mise au point et performances

Toutes les informations discutées ci-dessous concerne le coeur même de la gestion d'une base de données et donc, celles-ci ne sont accessibles que par l'administrateur.

3.5.1 Taille d'une base de données

Nous pouvons penser que la **taille d'une base de données** n'est limitée que par l'espace disque que la machine peut gérer, de même que plus les données prennent de la place sur disque, plus le temps de réponse pour une transaction sera important.

3.5.2 Données dynamiques d'utilisation de la base de données

Afin de permettre une utilisation optimum de la base de données, l'administrateur de celle-ci a accès à diverses tables du dictionnaire de données. Ces tables lui indiquent notamment quels sont les utilisateurs qui manipulent la base de données.

INFORMIX donne tous ces renseignements, mais ORACLE en fournit encore plus. En effet, l'administrateur peut également examiner quelles sont les manipulations réalisées sur la base de données et par quels utilisateurs elles le sont [UNIS3,88], [ORAC1,86].

3.5.3 Données statistiques d'utilisation de la base de données

Ces données, l'administrateur de la base de données peut les obtenir suite à une procédure de calcul qu'il peut éventuellement automatiser.

Ainsi, par exemple, ORACLE fournit à l'administrateur tous les éléments qui lui sont nécessaires pour calculer l'espace qu'occupe en mémoire les tables et index. De cette manière, il peut, suivant la vie de la

base de données, estimer l'espace disque actuel, futur, nécessaire à cette base de données [UNIS3,88], [ORAC1,86].

INFORMIX, quant à lui ne fournit malheureusement pas ce genre de renseignements.

3.5.4 Données pour mise au point et fonctionnement de l'optimiseur

Toujours dans le but d'avoir une base de données qui permette un travail performant, le système peut donner à l'administrateur une certaine liberté d'action.

Dans ce sens, ORACLE va permettre à cet administrateur de modifier des paramètres tels que, par exemple, la taille des tampons. Mais tous ces paramètres que seul l'administrateur peut manipuler ne sont seulement pris en compte que lors d'une initialisation ou réinitialisation de la base de données [ORAC1,86], [UNIS3,88].

INFORMIX, quant à lui ne donne pas une telle liberté d'action pour améliorer la performance de la base de données.

ORACLE dispose également d'un utilitaire appellé **ODS** et qui, lorsque des utilisateurs travaillent sur la base de données, permet d'examiner divers éléments comme : les différents processus utilisateurs qui utilisent à ce moment là ORACLE, les différents programmes ORACLE qui sont utilisés (SQL*PLUS, SQL*FORMS, ...), les tables qui sont accédées, les différents blocages de tables et/ou d'enregistrement qu'ORACLE réalise, le statut du fichier "Before Image", les différentes activités d'entrée/sortie. Ces données peuvent être très utiles lorsque l'on désire améliorer les performances de la base de données; ou lorsque l'on rencontre un problème, pour en déterminer la cause [ORAC1,86].

Suite à une requête, il se peut que le RDBMS doive parcourir un chemin à travers plusieurs tables. Afin de déterminer celui-ci de manière à obtenir un temps de réponse optimal, il fait appel à un optimiseur. Ce dernier va choisir un ordre de liaison entre les différentes tables indépendamment de l'ordre déterminé dans la requête.

Seul le fonctionnement de l'optimiseur d'ORACLE sera précisé (voir figure 10). Celui-ci va faire son choix en fonction de la construction des tables et/ou de la formulation de la requête. Dans ce tableau de la figure 10

, on peut voir dans quel ordre, et de quelle manière ORACLE examine les requêtes et les tables afin d'obtenir un parcours optimum entre les tables (plus le numéro d'ordre est petit, plus la réponse d'ORACLE est rapide) [ORAC1,86], [UNIS3,88].

Ordre d'examen	Elément examiné
1	- La requête contient le numéro de ligne de l'enregistrement dans le fichier = constante dans la condition
2	- La colonne d'une table a un index unique = constante dans la condition
3	- La colonne d'une table a un index unique concaténé = constante dans la condition
4	- Une clé d'une table "clustered" = une clé d'une autre table "clustered" mais ces deux tables font partie du même "cluster"
5	- La clé d'une table "clustered" = constante dans la condition
6	- La colonne d'une table a un index non unique concaténé = constante dans la condition
7	- La colonne d'une table a un index non unique = constante dans la condition
8	- Un index concaténé non comprimé >= limite
9	- Un index concaténé comprimé >= limite
10	- Un index concaténé non comprimé exprimé comme étant le plus important
11	- Un index concaténé comprimé exprimé comme étant le plus important
12	- Une colonne index unique comprise entre deux valeurs ou une colonne index unique comparable à un ensemble de valeurs
13	- Une colonne index non unique comprise entre deux valeurs ou une colonne index non unique comparable à un ensemble de valeurs
14	- Une colonne index unique < ou > à une constante
15	- Une colonne index non unique < ou > à une constante
16	- Une opération telle que SORT ou une jointure
17	- Les opération MAX ou MIN d'un simple colonne indexée
18	- L'opération ORDER BY où il y a un index
19	- Analyse de toute la table

Figure 10 : L'otpimiseur d'ORACLE

L'optimiseur va examiner séquentiellement ces éléments pour voir lequel d'entre eux correspond à la structure de la requête ou d'une table et il va arrêter son examen dès qu'il aura trouvé. Ainsi, le RDBMS peut prendre le chemin optimum pour le parcours à travers plusieurs tables et fournir une réponse en un temps minimum à l'utilisateur.

3.5.5 Réorganisation

Lorsque, malgré le travail de l'optimiseur, le temps d'accès à certaines tables semble excessivement important, même si cette table a une taille considérable, l'administrateur peut décider de la réorganiser sur disque. ORACLE, grâce à une instruction SQL va permettre ce genre d'opération [UNIS3,88], [ORAC4,86].

3.6 Résumé et conclusion

Dans ce chapitre 3, nous avons donc passé en revue quelques éléments de comparaison théorique. Bien entendu, ne sont repris ici que les plus importants et les plus intéressants. En résumé, nous pouvons exprimer en un tableau récapitulatif quels sont les points pour lesquels ORACLE et INFORMIX se valent, et ceux pour lesquels l'un semble meilleur que l'autre. Dans ce but, nous allons nous définir les sigles suivants :

- = : ORACLE et INFORMIX se valent;
- o : ORACLE semble meilleur qu'INFORMIX (oo : bien meilleur, ...);
- i : INFORMIX semble meilleur qu'ORACLE (ii : bien meilleur, ...);
- / : ni l'un, ni l'autre ne dispose de cette particularité.

Il ne faut bien sur pas perdre de vue que le tableau récapitulatif qui va suivre à la figure 11 apporte des conclusions qui ne tiennent encore aucunement compte des tests de performances qui seront expliqués dans les chapitres suivants.

INFORMATION GENERALES		ACCES POUR LES UTILISATEURS	
- La représentation des données .	o	- Le fonctionnement on line	=
- Le contrôle de l'intégrité	i	- Le fonctionnement depuis un langage hôte.	ii
- L'interrogation	i	- Le générateur d'écrans	i
- La modification des données ...	o	- Le générateur de rapports	i
PROTECTION		**MISE AU POINT ET PERFORMANCE**	
- La protection des accès	oo	- Le nombre de transactions	=
- Le chiffrement des données	/	- Les données d'utilisation dynamique	o
- Les accès multiples	o	- Les données d'utilisation satistique	o
- Le recouvrement et les back-up.	o	- Les données pour mise au point et fonctionnement de l'optimiseur	o
		- La réorganisation	=

Figure 11 : Récapitulatif théorique

Globalement, sur cette figure 11 nous remarquons qu'ORACLE s'impose plus qu'INFORMIX, et principalement au niveau de la protection où INFORMIX est très faible.

4. Principes théoriques sur la conception d'une base de données et sur les mesures de performances

4.1 Introduction

Quelques fondements théoriques nous semblent nécessaires et importants afin de bien cerner tous les éléments à prendre en considération lors de la réalisation pratique des chapitres 5 et 6. C'est pourquoi ce chapitre 4 va vous présenter quelques bases théoriques sur la conception d'une base de données ainsi que sur les mesures de performances.

4.2 Démarche générale de conception d'une base de données

Dans ce paragraphe, nous allons expliquer une méthode pour concevoir de manière systématique une base de données. Cette méthode a été développée en détail dans [HAINA,86] et nous ne reprendrons de cet ouvrage que les idées maîtresses.

4.2.1 Organisation et objectifs de la démarche

Cette méthode se réalise en trois phases : l'analyse conceptuelle, la conception logique et la conception physique. La figure 12 que l'on peut retrouver également dans [HAINA,86] nous montrent ces différentes étapes. Pour les deux premières phases, il existe un quadruple objectif:

- La solution finale doit être produite de manière systématique, respecter la spécification conceptuelle et être efficace pour les applications réalisées;

- Il faut essayer de retarder au maximum toutes les décisions concernant les langages de programmation et les **systèmes de gestions de données** (**SGD**) afin de minimiser l'effort à produire

lors d'un changement de **système de gestion de base de données (SGBD)** ou des caractéristiques physiques de la base de données;

- Les bases de données réalisées par cette méthode doivent pouvoir être gérées aussi bien par des logiciels spécialisés (SGBD) que par de simples systèmes de fichiers;

- Tout ce travail méthodique pourra servir à une programmation orientée base de données.

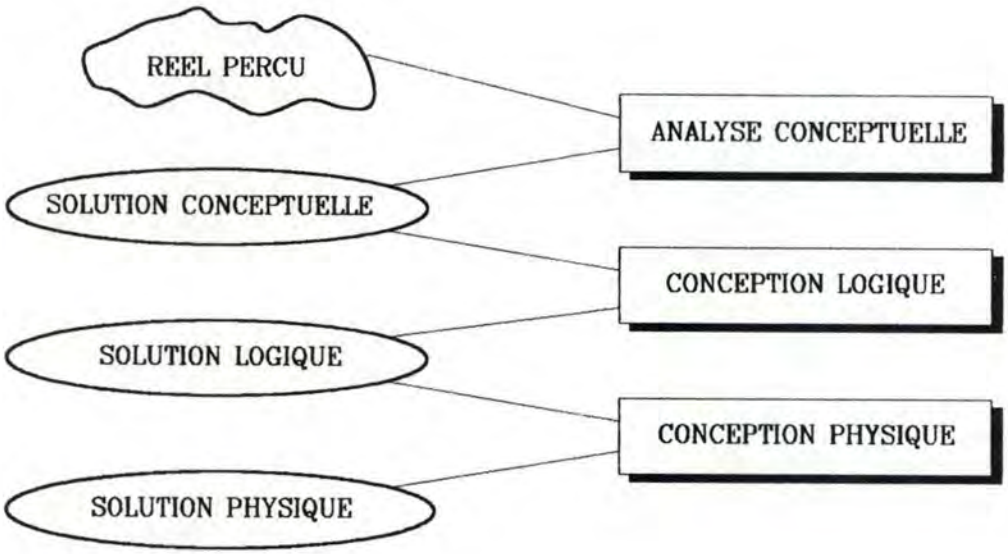


Figure 12 : Organisation générale de la démarche

4.2.2 L'analyse conceptuelle

Grâce à l'analyse conceptuelle, une description complète du système d'information est réalisée. Pour cela, il faut construire un **schéma conceptuel** des données, fournir une spécification des diverses fonctions qui devront être réalisées ainsi qu'un relevé quantitatif des données et des fonctions.

4.2.3 La conception logique

La conception logique s'inspire des résultats de l'analyse conceptuelle et donne comme résultat une solution exécutable par une machine abstraite. Cette solution logique doit être correcte (c'est-à-dire

qu'on doit y retrouver toute la sémantique du schéma conceptuel), efficace (on ne reprend que les mécanismes d'accès strictement nécessaires pour obtenir un accès optimisé aux données) et indépendante de la machine réelle. La solution exécutable par la machine abstraite se compose d'un **schéma MAG (Modèle d'accès généralisé)**et de modules (pour les fonctions) pour lesquels un algorithme **LDA (Langage de Description d'Algorithme)** a été développé.

Le schéma MAG s'obtient à partir du schéma conceptuel, il s'agit d'un schéma des accès possibles qui n'a d'autre but que de donner un formalisme nécessaire à la poursuite de l'étude.

Ensuite, à partir de la spécification conceptuelle des fonctions, des traitements, une architecture abstraite des modules est réalisée. Pour chaque module, on donne une spécification externe et un algorithme qui correspond à cette spécification. On parle alors d'**algorithmes prédicatifs**. Ces derniers peuvent être transformés en **algorithmes efficaces** (ou **algorithmes effectifs**) en optimisant les accès à la base de données. Pour réaliser cette optimisation, on tient compte du nombre d'opérations logiques.

Nous pouvons alors créer le **schéma des accès nécessaires**. Une des façons de construire celui-ci est de relever dans chaque algorithme effectif les données et les mécanismes d'accès utilisés. On fusionne alors tous ces relevés que l'on présente dans un schéma.

Une dernière opération à réaliser pour la conception logique est une carte exprimant l'**intensité du trafic**. Pour cela, à partir de la description statistique du schéma des accès possibles et grâce aux algorithmes effectifs, on établi une quantification dynamique. On va ainsi exprimer le nombre moyen d'activation par jour de chaque primitive, le nombre moyen d'éléments concernés par une activation et le produit de ces deux valeurs, ce qui représente le nombre moyen d'éléments activés par jour. Pour construire le schéma représentant l'intensité du trafic, on reprend ce nombre d'éléments activés par jour, et on le reporte à chaque type d'enregistrement représenté dans le schéma des accès nécessaires.

4.2.4 La conception physique

Grâce à la conception logique, la conception physique va fournir une solution physique. Celle-ci doit être correcte, efficace et exécutable.

Durant cette phase, beaucoup d'actions doivent être réalisées, voici les plus importantes.

Le premier élément à produire est un **schéma conforme au SGD**. Ce schéma est dérivé du schéma des accès possibles de la phase de conception logique et s'exprime sous le même formalisme MAG. Il est identique au schéma des accès possibles sauf qu'il tient compte des particularités, des possibilités du SGD choisi.

Ensuite, il s'agit de produire des **algorithmes effectifs conformes au SGD**. Comme dans l'étape précédente, les algorithmes effectifs de la phase de conception logique sont modifiés de manière à ce qu'ils prennent en compte les particularités et les possibilités du SGD.

Un autre étape importante de la phase de conception physique est l'**évaluation des consommations de ressources** que l'on obtient à partir du schéma des accès nécessaires. En effet, grâce à ce dernier, on peut déterminer les tailles des populations des types du schéma des accès nécessaires (nombre d'article de chaque type, nombre d'article par valeur de clé, ...). Ensuite, connaissant la taille de chaque objet (page, article de chaque type, ...), on peut définir les volumes des fichiers et/ou de la base de données ainsi que son taux de remplissage. Ces évaluations pourront aider à l'estimation des temps moyens d'exécution pour chaque opération élémentaire.

La suite de la conception physique consiste principalement en une suite d'opérations de traduction des différents résultats obtenus dans le langage de définition de données du SGD, de mise en route de la base de données, de tests de validation, de vérifications de performance ou encore de processus de protection contre les incidents.

La base de données est alors réalisée, et si une modification doit intervenir (au niveau de la structure des données, ou des applications par exemple), celle-ci, grâce à la méthode décrite succinctement ci-dessus, peut facilement être répercutée sans recommencer tout le travail.

4.3 Mesures de performances

Dans ce chapitre, nous allons préciser ce que l'on entend par "**benchmark**", et quelles sont les conclusions que l'on peut tirer suite à la réalisation de ces derniers.

4.3.1 Qu'est-ce qu'un "benchmark"

Un "benchmark" peut être défini comme un ensemble de tests qui permettent d'établir les performances d'un système d'exploitation, d'un logiciel, d'un réseau, ... (cette définition peut être retrouvée notamment dans [Illin,83]). Dans le cadre de ce travail, nous allons nous intéresser d'un peu plus près aux "benchmarks" relatifs aux bases de données.

Il existe dans la littérature des "benchmark" types tels que le **Debit/Credit** et son dérivé le **TP1**, le **Wisconsin (De Witt) Benchmark**, le **AS³AP Benchmark**, le **RAMP-C Benchmark**, le **1K (Onekay) Benchmark**, ... Tous ces "benchmarks" sont utiles lorsque l'on désire limiter le champ d'action d'un RDBMS à un but bien précis, mais lorsqu'il s'agit de tester les performances de sa propre application, on connaît les points précis à analyser qui sont en rapport direct avec l'application bien particulière, et dès lors la meilleure solution est de développer soi-même son propre "benchmark". C'est ces derniers que nous allons principalement étudier.

4.3.2 Quand décide-t-on de réaliser un "benchmark"?

La décision de réaliser un "benchmark" peut se prendre bien avant qu'une application n'existe, mais aussi pendant le cycle de vie de l'application. Grâce à [ISI4,89], on peut déterminer différents moments de réalisation d'un "benchmark" (cette liste n'étant pas exhaustive) :

- Lorsque l'on désire comparer plusieurs RDBMS en vue d'un achat;
- Lorsque l'on a des problèmes de performance, un "benchmark" peut être réalisé afin de trouver la solution optimale (garder la configuration actuelle ou l'étendre par exemple);
- Pour vérifier que les performances ne se dégradent pas;

- Pour identifier des causes de goulots d'étranglement éventuels du système qui diminuent très fortement les temps de réponse;

- Pour déterminer si les ressources du système sont suffisantes pour faire face au nombre croissant d'utilisateurs, ou pour estimer si une application peut s'implanter sur des sites de type différents.

4.3.3 Pièges à éviter lors de la réalisation d'un "benchmark"

Avant toute prise de mesures pour évaluer la performance d'un système, il faut déterminer un certain nombre d'hypothèses afin de refléter au mieux l'environnement réel que l'on désire tester.

La première chose à définir est donc le but des tests; en effet, selon qu'il s'agit de tester une application particulière sur une machine déterminée, ou une comparaison entre deux systèmes, la liberté d'action sera différente. Selon Nicolas Nierenberg [NIERE,86], lorsque l'on désire estimer les performances de DBMS au travers d'applications particulières, on peut facilement commettre des erreurs qui, même si les bonnes questions sont posées amènent une interprétation erronée des résultats. Voici quelques erreurs que cet auteur a relevées :

- Utiliser pour les tests un environnement machine dont les principales caractéristiques ne correspondent pas à celles du système sur lequel évoluera l'application dans le futur; ce système n'étant pas toujours disponible au moment des tests. Il faut donc dès lors trouver un environnement similaire afin d'obtenir des résultats crédibles;

- Lors des tests, à nouveau pour permettre d'obtenir des résultats qui soient valables, il s'agit de poser les bonnes questions; c'est-à-dire de réaliser des requêtes qui, au niveau du travail fournit par le RDBMS, soient comparables à celle de l'application que l'on désire tester, ou de l'application future possible;

- Afin d'avoir une simulation correcte d'une base de données pour un système, il faut non seulement que les requêtes utilisées pour le test reflètent un travail similaire pour le RDBMS que celles de l'application, mais il faut également que la taille de la base de

données du test reflète celle utilisée lors des applications futures. En effet, certains RDBMS sont très performants pour de petites bases de données, et d'autres pour un grand nombre de données. Ainsi, si la taille des données n'est pas simulée correctement, les interprétations des résultats pourraient être erronés;

- Lorsque l'on désire tester des entrées/sorties à partir de plusieurs terminaux, il est primordial de bien s'assurer de ne pas utiliser certaines facilités offertes par beaucoup de RDBMS et qui permettent le chargement de plusieurs enregistrements à la fois. En effet, dans ce cas, le test est complètement faussé;

- Lorsque la simulation d'une application est correcte et que l'on désire comparer deux RDBMS, il faut encore interpréter correctement les résultats c'est-à-dire ne pas rester limité à un seul point de vue tel que le nombre de transactions par seconde, mais regarder aussi la finalité des différentes transactions, déterminer les points les plus importants pour l'utilisateur. En fonction de ces données et du nombre de transactions par seconde, on peut affirmer quel sera le système le mieux adapté.

Lorsque l'on a pas d'application particulière à tester, mais que l'on désire simplement examiner les performances d'un ou plusieurs RDBMS, la liberté d'action est beaucoup plus grande. Il faut cependant, au départ se fixer des hypothèses et par la suite, dans la réalisation du "benchmark" et l'interprétation des résultats rester cohérent avec les hypothèses de départ et éviter également tous les pièges cités ci-dessus.

4.3.4 Réalisation de son propre "benchmark"

Afin d'éviter tous ces problèmes lorsque l'on réalise son propre "benchmark", Nicolas Nierenberg [NIERE,86] propose d'établir avant toute chose une analyse des besoins et de la charge au travers des deux grilles suivantes (figure 13 et 14) :

Définition des besoins des utilisateurs

Description des utilisateurs

- Type d'utilisateur
- Nombre d'utilisateur
- Mode d'accès
- Degré du partage des données
- Degré d'utilisation non planifiée
- Temps de réponse
- Accès et contrôle des données

Description des données

- Volume des données
- Croissance du volume des données
- Qualité des données
- Sensibilité à la perte ou aux dommages
- Taux de transactions

Exigences au niveau application

- Langage
- Documentation
- Portabilité

Exigences au niveau du système

- Mémoire
- Administration du système
- Coût
- Temps d'installation

Figure 13 : Définition des besoins des utilisateurs

Poids des différents besoins des utilisateurs

Besoins	Poids
Facilité pour des utilisateurs voulant accéder à la même information.....	
Facilité pour des bases de données importantes (40-60 MB).....	
Facilité pour des applications qui évoluent rapidement.....	
Portabilité sur différents environnements UNIX.....	
Facilité pour la définition des écrans.....	
Total.....	

Echelle des poids

- 0-2 : exigence faible
- 3-5 : exigence modérée
- 6-8 : exigence importante
- 9-10 : exigence primordiale

Figure 14 : Poids des différents besoins des utilisateurs

La première grille (figure 13) va nous permettre de bien mettre en évidence tous les paramètres du système à analyser; la deuxième quant à

elle (figure 14) va nous aider à définir la charge du système et à construire des requêtes, des transactions qui répondent aux finalités du RDBMS.

Mais avant de réaliser le "benchmark", il faut encore prendre en compte deux éléments :

- Bien que l'on connaisse les finalités du système, il faut encore préciser quels sont les caractéristiques, les éléments que l'on désire tester (par exemple le temps que met le RDBMS pour réaliser une transaction bien particulière);

- Les tests peuvent se limiter aux seuls éléments dont on parle dans le point précédent, mais l'application va vivre, et suite à cela, les performances pourraient se dégrader. Dès lors, afin de choisir avec pertinence quel environnement, ou quel RDBMS utiliser, on peut essayer de faire varier certains paramètres afin de simuler les évolutions possibles de la base de données (augmentation du nombre d'utilisateurs, augmentation du volume des données à traiter, ...) et ainsi voir les répercussions possibles au niveau des performances. Si les résultats de ces tests ne sont pas satisfaisants, on pourra modifier des paramètres du RDBMS quand cela est possible, ou encore modifier des paramètres du système d'exploitation pour retrouver des performances acceptables.

Il faut donc déterminer les paramètres qui pourraient évoluer dans le futur; dans [ISI4,89] on peut retrouver les paramètres suivants :

- Le nombre d'utilisateurs simultanés de l'application;

- La composition des requêtes, des transactions;

- La vitesse à laquelle les transactions sont envoyées au RDBMS; c'est-à-dire une variation de la charge;

- La taille et la structure de la base de données; c'est-à-dire s'il y a des index, une manière particulière de ranger physiquement les données,...

- La configuration système.

4.3.5 Traitement et analyse des résultats

Lorsque les mesures ont été réalisées, la masse de résultats obtenus doit être présentée de manière lisible (le plus souvent sous forme de graphe) afin d'en permettre rapidement et facilement l'interprétation.

Ainsi que cela a été précisé dans le paragraphe précédent, à tout moment on peut obtenir des résultats qui ne sont pas satisfaisants; dès lors, il faut essayer de modifier des paramètres du RDBMS ou du système d'exploitation afin de retrouver un niveau de performance acceptable. Ainsi, dans [FERRA,83] on peut trouver divers facteurs qui peuvent influencer les performances d'un RDBMS :

- L'organisation physique de la base de données;
- Les caractéristiques des périphériques (disques, bandes magnétiques, ...);
- Taille de la mémoire réservée pour les tampons;
- Le nombre de demandes d'entrée/sortie pour chaque transaction individuelle;
- Les opérations nécessaires pour maintenir l'intégrité de la base de données (création de point de synchronisation, création de fichiers pour le "journalling", ...);
- Des paramètres propres au RDBMS (par exemple le nombre maximum de processus concurrents propre au RDBMS, le nombre maximum d'utilisateurs en mémoire cache, ...);
- Des paramètres propres au système d'exploitation (par exemple le nombre maximum de processus concurrents).

4.4 Résumé et conclusion

Dans ce chapitre, nous avons vu comment construire de manière systématique, en trois phases (analyse conceptuelle, conception logique, conception physique), une base de données.

Le paragraphe suivant quant à lui nous a donné quelques informations au sujet de la création de son propre "benchmark" : quelques pièges à éviter, des grilles d'analyse permettant de cerner un problème, l'établissement d'hypothèses et de paramètres qui simulent l'évolution du système.

Ce chapitre théorique va nous servir d'introduction aux chapitres 5 et 6 qui sont consacrés à l'élaboration d'une base de données et à diverses mesures de performances sur celle-ci.

5. Elaboration de la base de données

5.1 Introduction

Dans ce chapitre, nous allons présenter la création de la base de données testée. Celle-ci pourrait correspondre à un problème réel, mais elle n'a été développée que dans le but de produire des mesures de performances. La démarche utilisée sera celle qui a été exposée dans le chapitre précédent, au paragraphe 4.1. Etant donné que le but de ce travail est l'analyse des performances, et non pas la conception et la création d'une base de données, nous n'illustrerons pas tout ce qui a été présenté dans cette partie théorique, mais uniquement ce que nous jugeons utile pour la bonne compréhension et la poursuite de l'analyse.

5.2 Conception systématique de la base de données

5.2.1 Analyse conceptuelle

Nous allons tout d'abord proposer un énoncé en français de la base de données désirée. Soit un centre sportif. Ce centre propose un certain nombre de sports différents et pour chaque sport, on dispose d'un club. Ce centre dispose également de membres et chaque membre a la possibilité de pratiquer un ou plusieurs sports. Pour chaque sport pratiqué, une licence est exigée. Lorsqu'un membre désire pratiquer un de ses sports favoris, il doit faire une réservation de terrain. Dans ce centre, chaque club rencontre des clubs adverses en matches amicaux ou de championnat. Le résultat de toute rencontre sera retenu, ainsi que la date de cette rencontre. Cette dernière est identifiée par un numéro (NRC) et si un match est reporté, cela est indiqué. Le centre sportif est identifié par un numéro (NCS), il a un nom, une adresse et un numéro de téléphone. Un club est identifié par un numéro (NC), il a un nom, une adresse et un numéro de téléphone. Un club adverse possède les mêmes caractéristiques. Un membre

est identifié par un numéro de membre (NM), il a un nom, un prénom, une adresse, un numéro de téléphone (non obligatoire) ainsi qu'une date de naissance. Toute licence porte un numéro unique (NL) ainsi qu'un montant de cotisation. Lorsqu'un membre fait une réservation, il doit indiquer un numéro de salle, un numéro de terrain, une date et une heure. Une réservation porte également un numéro unique.

Cet énoncé peut être transformé en un schéma conceptuel qui est représenté à la figure 15.

Ensuite, nous donnons la liste des traitements possibles pour cette base de données avec différents utilisateurs potentiels possibles. Soit quatre types d'utilisateurs ayant chacun la possibilité de réaliser des requêtes différentes :

1. Un simple membre a la possibilité de :

- Consulter une liste (de membres d'un club donné, de résultats à une date précise, de réservations de terrain à une date donnée);
- Ajouter une réservation de terrain;
- Supprimer une réservation de terrain;
- Modifier une réservation de terrain.

2. Le trésorier a toutes les possibilités d'un simple membre, et de plus, il pourra exécuter les fonctions suivantes :

- Ajouter un nouveau membre et sa licence correspondante;
- Supprimer un membre et ses licences;
- Modifier les caractéristiques d'un membre ou de ses licences;
- Modifier le montant des cotisations.

3. Un responsable de club, a tous les privilèges d'un simple membre, et de plus a les possibilités suivantes :

- Ajouter un nouveau résultat;
- Modifier des résultats.

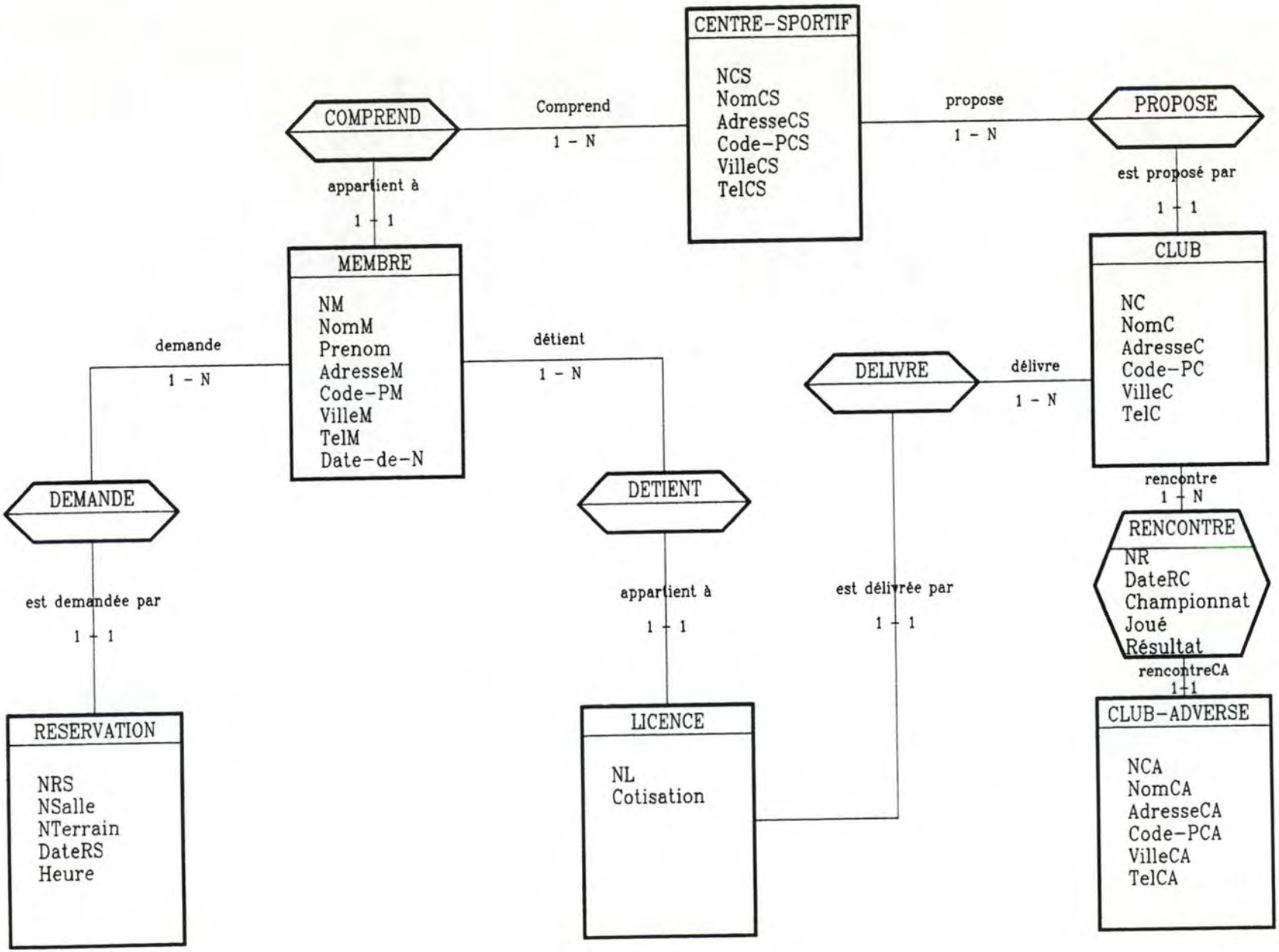


Figure 15 : Schéma conceptuel de la base de données

4. Le responsable du centre possède quant à lui tous les privilèges et il peut en outre :

- Modifier les caractéristiques du centre sportif;
- Ajouter un nouveau club ou un nouveau club adverse;
- Supprimer un club ou un club adverse;
- Modifier les caractéristiques d'un club ou d'un club adverse.

On propose alors de créer un module par type d'utilisateursutile; on aurait donc quatre modules, le quatrième ayant la possibilité d'utiliser tous les autres.

D'un point de vue quantitatif maintenant, on estime qu'il y a 10.000 membres répartis dans 15 clubs. Chaque membre fait partie en moyenne de 2 clubs, il y a donc 20.000 licences. Les clubs adverses quant à eux sont au nombre de 90.

D'un point de vue quantitatif au niveau des fonctions :

- Un membre fait en moyenne 4 réservations par mois et en supprime 1 tous les 2 mois. Il modifie une réservation 1 fois tous les 2 mois également. Un membre consulte une liste de résultats 4 fois par mois, la liste des membres d'un club 1 fois tous les 2 mois et la liste des réservations 4 fois par mois;
- Le trésorier crée en moyenne 20 nouveaux membres par mois (ce qui correspond environ à 16 nouveaux membres par club par an). Il en supprime environ 8 par mois, modifie 1600 cotisations par mois et il change les caractéristiques d'un membre 2 fois par mois;
- Le responsable de club quant à lui ajoute en moyenne 4 nouveaux résultats par mois et en modifie 1 tous les deux mois;
- Le responsable du centre crée un nouveau club 1 fois tous les deux ans et un club adverse 3 fois par an. Il supprime un club adverse 2 fois par an, et un club propre, pratiquement jamais. Quant à la modification des caractéristiques d'un club ou d'un club adverse, cela se fait environ 1 fois par an. Ce responsable modifie aussi les caractéristiques du centre sportif, et ce 1 fois

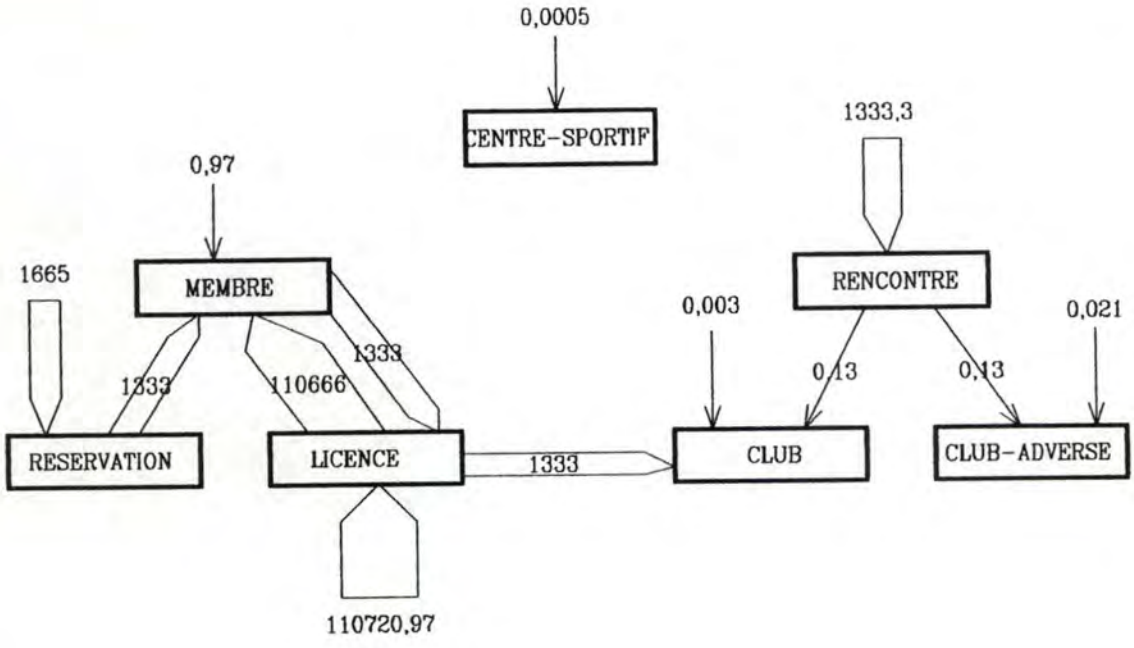


Figure 16 : Intensité du trafic

tous les 5 ans (un numéro de téléphone supplémentaire, changement de code postal par exemple).

5.2.2 Conception logique

Pour cette conception logique, nous allons tout d'abord élaborer le schéma MAG (voir figure 17) correspondant au schéma conceptuel.

Nous n'allons pas développer d'algorithme prédicatif pour toutes les requêtes proposées lors de la phase d'analyse conceptuelle; mais nous reprendrons ultérieurement quelques requêtes représentatives pour les mesures de performances.

Par contre, dans le tableau de la figure 18, nous pouvons voir toutes les estimations statistiques transformées en **nombre d'activations par jour** d'une primitive **(NA/J)**, en **nombre d'éléments concernés par une activation (NE/A)** et on en déduit le **nombre d'éléments traités par jour (NE/J)**.

Nous n'exprimerons pas le schéma des accès nécessaires. Cependant, nous allons élaborer, sur base du tableau de la figure 18, le schéma représentant l'**intensité du trafic** (voir figure 16), ce qui nous sera plus utile pour l'analyse ultérieure.

5.2.3 Conception physique

Pour la conception physique, nous allons essentiellement présenter le **schéma MAG conforme à un RDBMS** (voir figure 19).

Nous ne parlerons pas non plus d'algorithmes effectifs conformes aux RDBMS car, dans la suite du travail, pour réaliser les mesures de performances, nous présenterons des requêtes conformes à ORACLE et INFORMIX.

Quant à l'évaluation des consommations des ressources, celle-ci sera en partie présentée au paragraphe 5.4 de ce même chapitre lorsque la taille résultante de la base de données sous les deux RDBMS sera évaluée.

Pour terminer ce paragraphe concernant la conception physique, nous allons présenter dans le tableau de la figure 20 les types de données qui ont été assignés aux champs des enregistrements des tables des deux RDBMS. Ces types de données ne sont pas tout à fait identiques. En effet, des particularités des SGBD ont été utilisées comme le type "serial" pour INFORMIX (ce type de données avait été présenté au point 3.1.2.3 du chapitre 3).

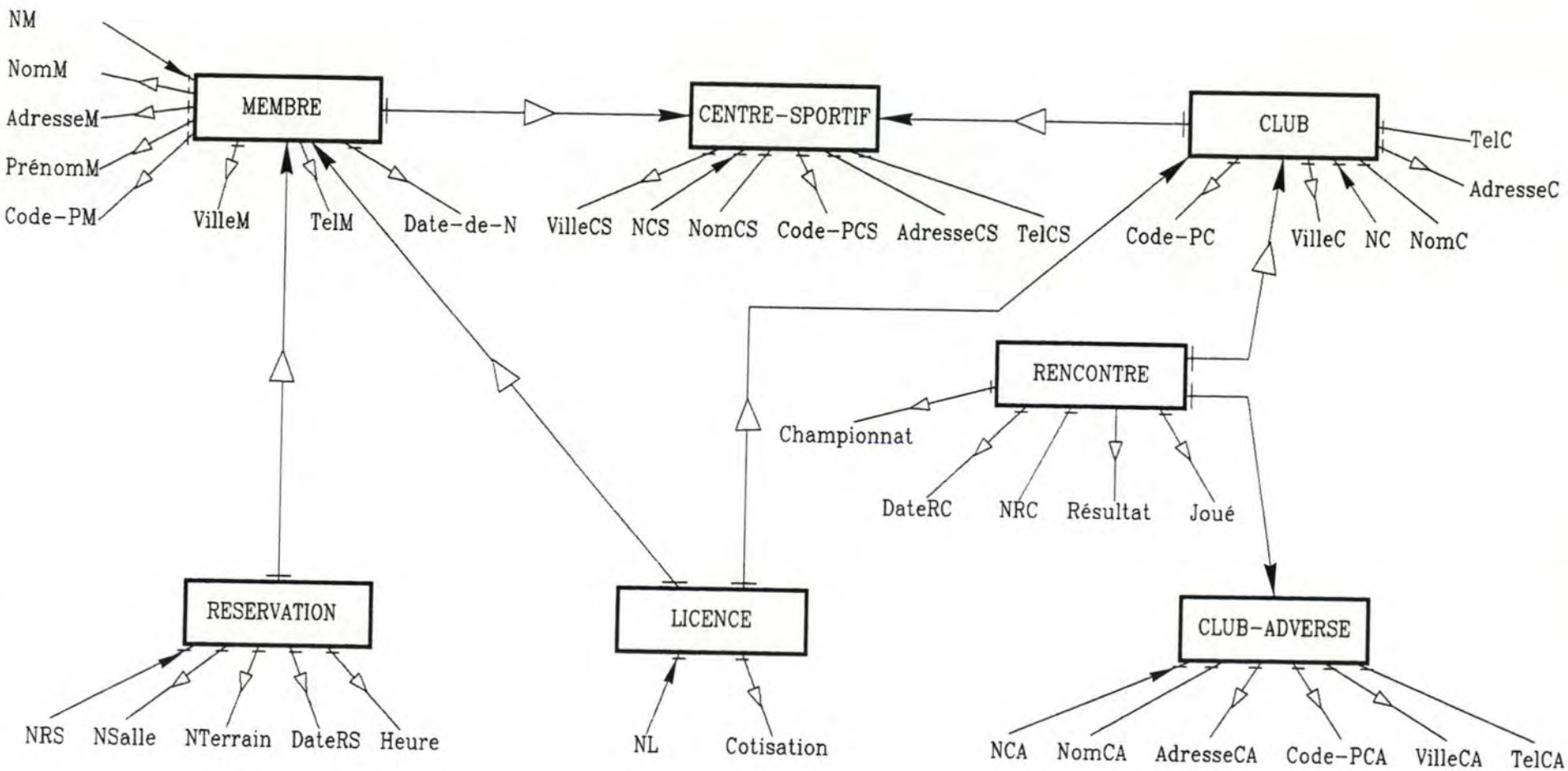


Figure 17 : Schéma MAG de la base de données

	NA/J	NE/A	NE/J
MEMBRES			
- Créer une réservation	1333	4	5332
- Modifier une réservation	166	1	166
- Supprimer une réservation	166	1	166
- Consulter réservation (date = X)	1333	1	1333
- Consulter les membres (club = X)	166	1333	221278
- Consulter les résultats (date = X)	1333	1	1333
TRESORIER			
- Créer un membre et ses licences	0.6	2	1.2
- Supprimer un membre et ses licences	0.3	2	0.6
- Modifier les caractéristiques d'un membre	0.07	2	0.14
- Modifier les montants des cotisations	54	1	54
RESPONSABLE DE CLUB			
- Ajouter un nouveau résultat	0.13	3	0.39
- Modifier un résultat	0.17	1	0.17
RESPONSABLE DU CENTRE SPORTIF			
- Créer un club	0.001	1	0.001
- Créer un club adverse	0.014	1	0.014
- Supprimer un club	/		
- Supprimer un club adverse	0.005	1	0.005
- Modifier les caractéristiques d'un club	0.002	1	0.002
- Modifier les caractéristiques d'un club adverse	0.002	1	0.002
- Modifier les caractéristiques du centre sportif	0.0005	1	0.0005

Figure 18 : Quantification statistique des primitives

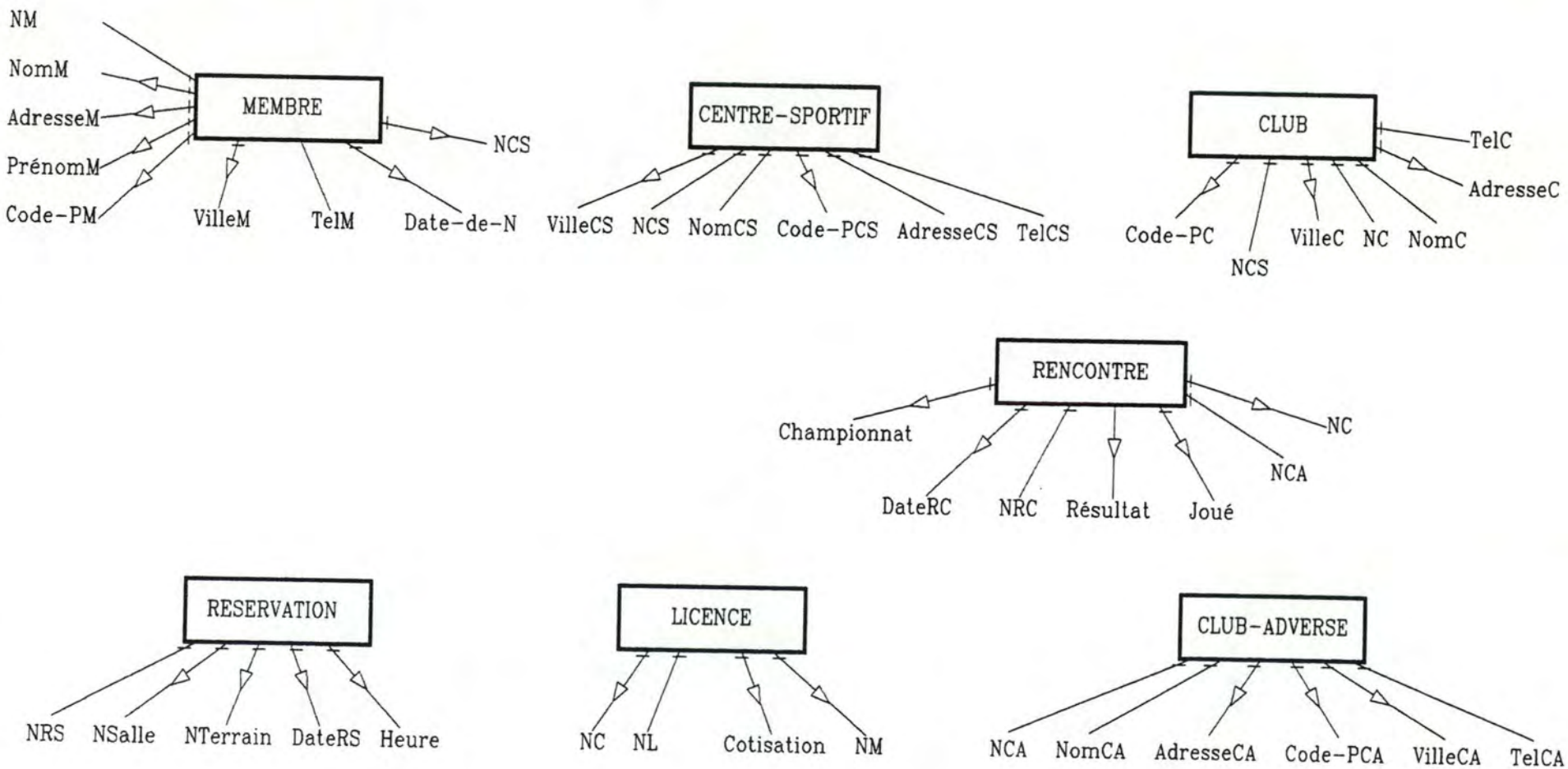


Figure 19 : Schéma conforme RDBMS

TABLES	CHAMPS	INFORMIX	ORACLE
CENTRE-SPORTIF	NCS	serial (1)	number
	NomCS	char (20)	char (20)
	AdresseCS	char (20)	char (20)
	Code-PCS	integer	number
	VilleCS	char(15)	char(15)
	TélCS	char (18)	char (18)
MEMBRE	NM	serial (1)	number
	NomM	char (15)	char (15)
	Prénom	char (10)	char (10)
	AdresseM	char (15)	char (15)
	Code-PM	integer	number
	VilleM	char (9)	char (9)
	TélM	char (12)	char (12)
	Date-de-N	date	date
	NCS	integer	number
LICENCE	NL	serial (1)	number
	Cotisation	integer	number (5)
	NC	integer	number
	NM	integer	number
RESERVATION	NRS	serial (1)	number
	Nsalle	integer	number
	Nterrain	integer	number
	Date	date	date
	Heure	decimal (2,2)	number (2,2)
	NM	integer	number
CLUB	NC	serial (1)	number
	NomC	char (20)	char (20)
	AdresseC	char (20)	char (20)
	Code-PC	integer	number
	VilleC	char (15)	char (15)
	TélC	char (18)	char (18)
	NCS	integer	number
RENCONTRE	NRC	serial (1)	number
	Date	date	date
	Championnat	char (1)	char (1)
	Joué	char (1)	char (1)
	Résultat	char (25)	char (25)
	NC	integer	number
	NCA	integer	number
CLUB-ADVERSE	NCA	serial (1)	number
	NomCA	char (20)	char (20)
	AdresseCA	char (20)	char (20)
	Code-PCA	integer	number
	VilleCA	char (15)	char (15)
	TélCA	char (18)	char (18)

Figure 20 : Les types de données

5.3 Principe d'élaboration et de remplissage automatisé de la base de données

5.3.1 Principe général

La création et la destruction automatique de la base de données, aussi bien en **INFORMIX-4GL** qu'avec **SQL*PLUS** d'ORACLE se font via un menu général écrit en shell UNIX. Ce menu permet également l'examen de différents temps de création (création de tables, chargement de tables, ...). Ces derniers seront examinés lors de l'analyse des résultats des mesures, au chapitre 6.

L'option de création du menu fait appel à un script général, écrit également en shell UNIX et qui enchaîne les différent modules 4GL d'INFORMIX ou SQL d'ORACLE, ainsi que des procédures UNIX qui créent des enregistrements et les mettent sous un format qui va permettre leur chargement dans les différentes tables de la base de données. Certains enregistrements devant être générés en grande quantité (10.000 ou 20.000), des jointures multiples entre plusieurs tables comprenant chacune quelques enregistrements de 1 caractère ou de 1 entier on été réalisées. Les enregistrements sont donc créés via des fichiers ASCII sous un format qui permet leur chargement dans les différentes tables grâce à un utilitaire de chargement de données (DBLOAD pour INFORMIX et ODL pour ORACLE). Le format requis par ces deux utilitaires n'est pas identique et sera expliqué aux points 5.3.3 et 5.3.4.

L'option de destruction de la base de données du menu principal consiste pour INFORMIX en une destruction simple de fichiers; car lorsqu'INFORMIX crée une base de données, il crée pour celle-ci un répertoire qui lui est réservé. Pour ORACLE, cette destruction se fait via un module SQL qui comprend des commandes de destruction de tables et des index.

Dans l'annexe 1, on peut trouver le script présentant le menu principal ainsi que ceux nécessaires à la création et à la destruction des bases de données d'INFORMIX et d'ORACLE.

5.3.2 Chargement sous INFORMIX

L'utilitaire **DBLOAD** (**Load Utility INFORMIX**, version 2.10) permet de charger un fichier ASCII dans une table de la base de données. Avant le chargement, cette table doit déjà exister (même si elle ne contient encore aucune donnée).

Pour pouvoir s'exécuter, le DBLOAD a besoin au moins de trois éléments :

- Le nom de la base de données dans laquelle sont définies la ou les tables à charger;
- Le nom d'un fichier de commandes;
- Le nom d'un second fichier dans lequel il indique les erreurs de chargement si il y a lieu.

Le fichier de commandes requis possède un format spécial, et il demande les informations suivantes :

- Le nom du fichier dans lequel se trouvent les données à charger;
- Le nom de la table dans laquelle les données doivent être chargées;
- Les noms des champs dans lesquels doivent être placées ces données.

Avec un même fichier de commandes, plusieurs tables peuvent être chargées.

5.3.3 Chargement sous ORACLE

Tout comme INFORMIX, ORACLE dispose de son utilitaire de chargement de données dans des tables à partir d'un fichier ASCII. Il s'agit de **ODL** (**ORACLE Data Loader** version 5.1.22). Avant de pouvoir utiliser ODL, les tables qui reçoivent les données doivent avoir été créés.

Pour pouvoir s'exécuter, ODL demande au minimum :

- Le nom d'un fichier de commandes (appelé le control file);

- Le nom d'un fichier dans lequel il résume ce qui s'est passé durant le chargement (le nombre d'enregistrements lus dans le fichier ASCII, le nombre d'enregistrements écrits dans la table, le nombre d'enregistrements rejetés);

- Un nom d'utilisateur avec son mot de passe.

Dans le fichier de commandes, on doit retrouver les informations suivantes :

- La définition des champs (nom et type de données) d'un enregistrement du fichier ASCII à charger;

- Le nom du fichier ASCII contenant ces enregistrements ainsi que la longueur d'un enregistrement;

- La table et les champs de ces enregistrements dans lesquelles les données seront chargées.

A l'opposé d'INFORMIX, cet utilitaire d'ORACLE ne permet pas de charger plusieurs tables à partir d'un même fichier de commandes. Par conséquent, pour chaque table que l'on désire charger, un fichier de commandes doit être créé.

5.3.4 La hiérarchisation des modules

Les deux paragraphes précédents permettent d'expliquer l'énorme différence qui existe dans la découpe modulaire (physique) de la création des deux bases de données. En effet, puisque ORACLE ne permet pas de charger plusieurs tables avec un même fichier de commandes, chaque fois qu'une table intermédiaire doit être chargée (ces tables servent à la création des enregistrements des tables MEMBRE et LICENCE par multiples jointures), un fichier de commande est créé. A l'annexe 2, on peut observer cette différence dans la modularisation physique.

5.4 Taille résultante sous les deux langages

Ainsi que cela a été expliqué dans le chapitre 3 (au point 3.4.3), ORACLE fournit la possibilité de calculer l'espace mémoire nécessaire pour

ses tables et index. Nous allons dès lors présenter un tableau de synthèse (figure 21) qui va reprendre la taille de la base de données sous différentes hypothèses. Ces dernières seront plus amplement expliquées et exploitées lors des mesures de performances analysées au chapitre 5.

	Nb de blocs de 1024 Bytes	Nb de Bytes
- Base de données de taille 1	1858	1.902.592
- Index de la base de données de taille 1	555	568.320
- Base de données de taille 1 avec 2 tables clustered	2028	2.076.672
- Base de données de taille 2 avec 2 tables clustered	4028	4.124.672
- Index de la base de données de taille 2	913	934.912

Figure 21 : Taille de la base de données ORACLE

Pour INFORMIX, la seule façon de calculer la taille de la base de données est de regarder le nombre de blocs que va prendre l'ensemble des fichiers dans le répertoire qui lui est attribué. Il est assez remarquable de noter que lorsque l'on ajoute des index, l'espace disque utilisé pour le répertoire ne change pas, comme si INFORMIX calculait l'espace nécessaire lors du remplissage des tables et que cet espace est réservé, même si la création d'index n'est pas demandée.

Pour INFORMIX, nous aurons donc dans le tableau récapitulatif de la figure 22 seulement deux hypothèses, à savoir les deux tailles de base de données (ces deux tailles en nombre d'enregistrements étant bien entendu identiques à celles d'ORACLE).

	Nb de blocs de 512 Bytes	Nb de Bytes
- Base de données de taille 1	3794	1.942.528
- Base de données de taille 2	5438	2.784.256

Figure 22 : Taille de la base de données INFORMIX

5.5 Résumé et conclusion

Ce chapitre nous a donc présenté la conception de la base de données selon les principes qui avaient été discutés théoriquement au point

4.1 du chapitre précédent; à savoir l'analyse conceptuelle, la conception logique et la conception physique.

Ensuite, la réalisation de la base de données a été proposée et l'on a pu remarquer que la conception physique de la base de données INFORMIX, au niveau des modules, semble plus simple. Quant à la taille des bases de données, on ne peut comparer que la taille index compris, et celle-ci est plus importante pour ORACLE que pour INFORMIX.

6. Mesures de performances réalisées

6.1 Principe général et requêtes utilisées

Le "**benchmark**" qui a été réalisé avait donc un triple but : tester ORACLE, tester INFORMIX et comparer ces deux RDBMS. Et c'est dans ce but que l'application développée dans le chapitre 5 a été créée.

Etant donné que nous avons voulu tester le **SQL** d'ORACLE et le **4GL** d'INFORMIX, aucun langage hôte n'a été utilisé, mais toutes les procédures ont été écrites en SQL ou 4GL.

De simples requêtes ont été créées pour réaliser ces tests; il s'agit :

- d'**ajout** : dans une table (la table RESERVATION), une création d'enregistrement a été réalisée;
- de **suppression** : dans une table (la table RESERVATION), une suppression d'enregistrement a été réalisée;
- de **mise à jour** : dans la table LICENCE, un enregistrement choisi aléatoirement a vu son champ numérique "cotisation" augmenté d'une unité;
- de **sélection simple** : il s'agit d'une simple jointure entre deux tables (les tables MEMBRE et LICENCE) pour retrouver un ensemble d'enregistrements;
- de **sélection plus complexe** : il s'agit toujours de la même jointure entre ces deux tables, mais des conditions supplémentaires ont été ajoutées, à savoir un opérateur "or", une condition "ORDER BY".

Tous les codages de ces requêtes se trouvent en annexe 3.

En ce qui concerne les prises de temps, celles-ci ont toujours été effectuées (grâce à des procédures écrites en langage C) le plus près possible avant et après l'exécution de la requête. Il faut toutefois noter que le temps d'exécution étant mesuré en secondes, et l'exécution des requêtes étant très rapide, il a fallu mesurer ces temps d'exécution pour un ensemble de requêtes d'un même type. Pour permettre l'analyse ultérieure, il faut ajouter que toutes ces mesures ont été transformées en nombre de **transactions par seconde** et c'est donc en terme de **TPS** que nous allons procéder à l'analyse.

Au chapitre 4, en présentant la partie théorique sur le "benchmark", nous avons vu au point 4.2.4 une manière de réaliser son propre "benchmark". Dans le cas qui nous occupe, nous n'allons pas passer en revue tous les éléments des deux grilles d'analyse que propose Nicolas Nierenberg [NIERE,86], mais lors des points suivants, nous détaillerons toutes les hypothèses qui ont été élaborées concernant l'état des deux RDBMS.

Pour toutes les hypothèses qui simulent en quelque sorte des événements possibles qui pourraient survenir dans la vie du système, nous avons voulu examiner les performances quand le nombre d'utilisateurs augmentait régulièrement.

Pour simuler cette augmentation d'utilisateurs, plusieurs procédures ont été lancées en mode "**batch**"; chacune d'elles représentant un utilisateur. Toutes ces mesures ont été prises en "**single user**" et durant la nuit pour éviter le plus de perturbations possibles et essayer de retrouver l'environnement dans un état plus ou moins identique à chaque prise de mesures.

Par la suite, pour ORACLE et INFORMIX, nous allons étudier les résultats par type de requête et par type d'hypothèse et pour la comparaison entre les deux, nous regarderons une à une les requêtes sous les différentes hypothèses.

6.2 Résultats obtenus et analyse théorique

6.2.1 Pour ORACLE

6.2.1.1 Hypothèses

Pour faire cette analyse, nous avons donc développé différentes hypothèses :

- **Hypothèse 1** : la taille de la base de données est celle qui a été définie lors de l'élaboration de celle-ci au chapitre 4 lorsque nous avons donné quelques éléments de quantification. Toutes les requêtes ont été réalisées sans la création d'index. Tous les paramètres d'ORACLE qui peuvent être modifiés lors de l'initialisation gardent leur valeur par défaut;

- **Hypothèse 2** : nous reprenons les même paramètres que pour l'hypothèse 1, mais dans ce cas, pour chaque valeur de clé d'accès, nous avons créé un index. De plus, pour faciliter la sélection, un autre index a été créé pour la table LICENCE (index sur le numéro de club);

- **Hypothèse 3** : nous reprenons dans ce cas tous les éléments de l'hypothèse 2, mais en modifiant un paramètre à l'initialisation. Il s'agit du nombre de tampons de données que ORACLE se réserve en mémoire cache. Ce nombre passe de 50 (valeur par défaut) à 250.

- **Hypothèse 4** : dans ce cas, nous reprenons toujours les paramètres de l'hypothèse 2 mais les deux tables pour lesquelles une jointure est réalisée, prennent un format spécial, elles sont "**clustered**". C'est-à-dire que d'un point de vue physique, ces tables ont une organisation particulière. Soit on sait qu'un membre possède en moyenne deux licences. On décide alors de créer un "cluster" dont la clé est le numéro de membre. Dans ce cas, toutes les licences correspondant à un même numéro de membre sont rangées physiquement "près" de l'enregistrement correspondant à ce membre.

- **Hypothèse 5** : il s'agit de la même hypothèse que l'hypothèse 4, mais on a augmenté la taille de la base de données. La table MEMBRE comprend 15.000 enregistrements (au lieu de 10.000) et la table licence comprend quant à elle 30.000 enregistrements au lieu de 20.000). La base de données a donc une taille 2.

6.2.1.2 Résultats par type de requête pour les différentes hypothèses

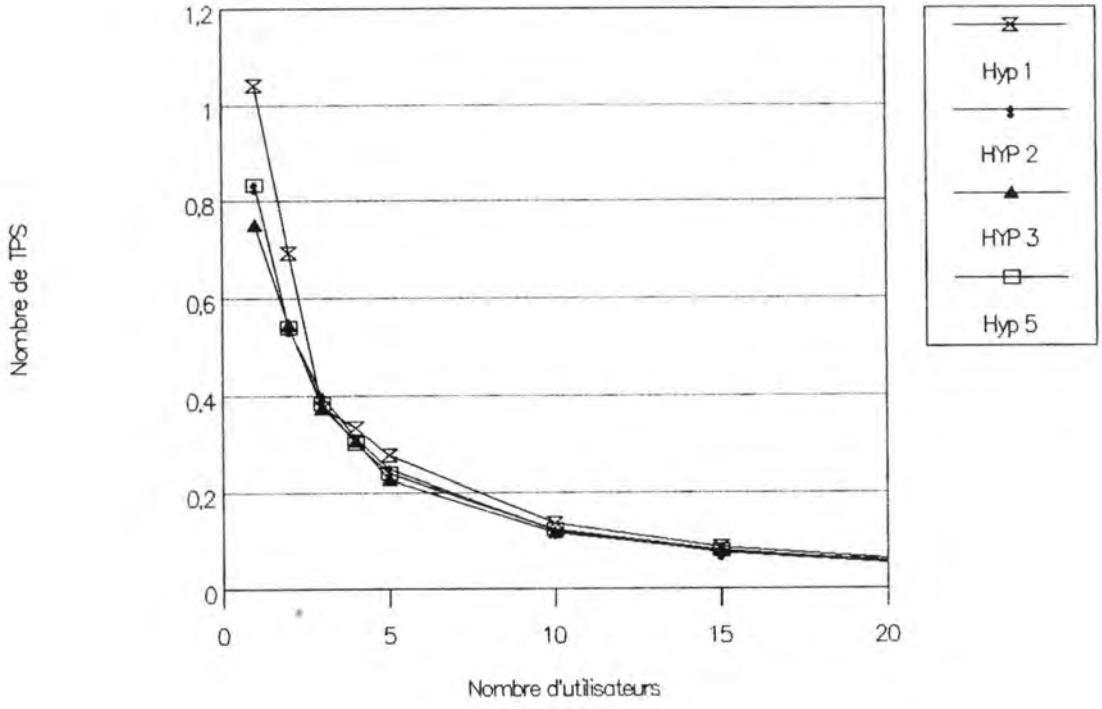


Figure 23 : ORACLE : Ajout

A la figure 23, nous pouvons observer l'évolution du nombre de TPS pour l'ajout d'enregistrements pour toutes les hypothèses. L'hypothèse 4 n'est pas représentée car la table dans laquelle la création d'enregistrements est réalisée n'est pas "clustered", et dès lors, l'hypothèse 2 et l'hypothèse 4 deviennent équivalentes. L'hypothèse 5 reste valable quant à elle puisqu'elle concerne une base de données plus importante. Nous pouvons observer une assez grande homogénéité dans les courbes et constater également, comme on pouvait s'y attendre, que l'absence d'index favorise cette création. Pour les autres courbes, la différence est trop peu significative que pour en tirer une conclusion valable.

Pour la figure 24 qui reprend l'évolution du nombre de TPS pour la suppression d'enregistrements, la même remarque que celle exprimée pour la figure précédente au sujet des hypothèses 4 et 5 peut être faite. Dans ce cas-ci, on peut voir que l'index n'a absolument aucune influence. L'ensemble des courbes semblent se confondre et la seule différence que l'on peut observer se situe au niveau du cas mono utilisateur où l'hypothèse la plus défavorable semble être celle pour laquelle une augmentation du nombre de tampons de données ORACLE a été réalisée, c'est-à-dire l'hypothèse 3. Toutefois, cette différence semble peu significative.

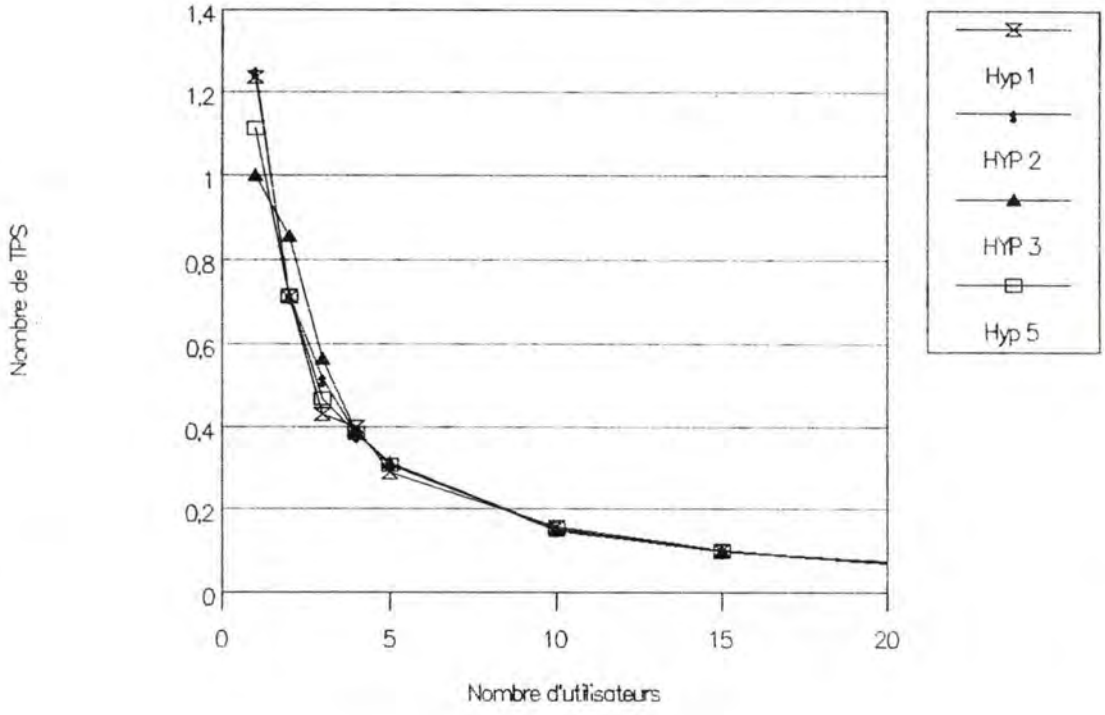


Figure 24 : ORACLE : Suppression

En ce qui concerne l'évolution du nombre de TPS pour la mise à jour, celle-ci semble plus intéressante. En effet, à la figure 25, nous pouvons voir que la présence d'index semble amener une petite amélioration. Mais le fait le plus remarquable est observé pour l'hypothèse où les tables MEMBRE et LICENCE sont "clustered". Dans ce cas, le fait que la taille de la base de données augmente un peu ne produit aucun effet. Pour la deuxième hypothèse, nous remarquons une anomalie au niveau de trois utilisateurs où le nombre de TPS est supérieur à celui de deux utilisateurs. Cette

bizarrerie est un élément connu pour ORACLE, bien que restant encore inexpliqué. Quant à l'hypothèse numéro 3, elle donne les plus mauvais résultats.

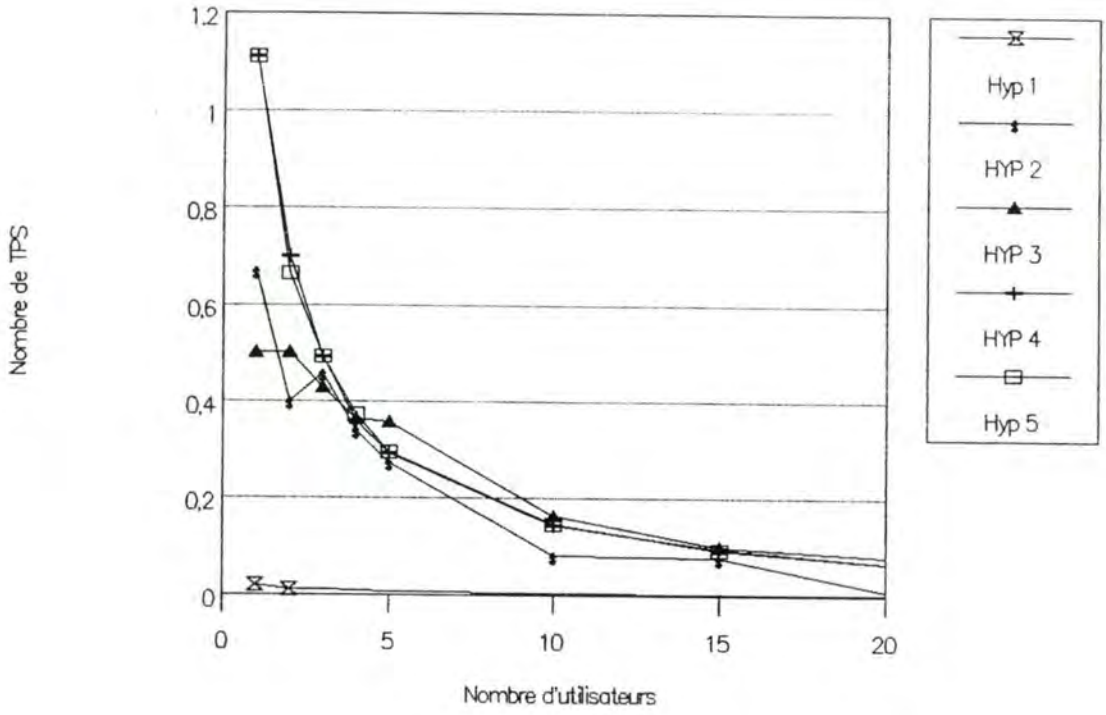


Figure 25 : ORACLE : Mise à jour

Le cas de la figure 26 nous présente l'évolution du nombre de TPS pour la sélection simple. Une fois encore, nous pouvons observer que le phénomène du "clustering" amène de bons résultats par rapport aux autres hypothèses. Et l'hypothèse 3 nous donne toujours les moins bons résultats.

Pour la sélection complexe, à la figure 27, nous retrouvons encore les même conclusions que pour la sélection simple. Le fait d'augmenter la taille de la base de données ou de modifier le nombre des tampons de données ORACLE semble quant à eux produire beaucoup moins d'effets.

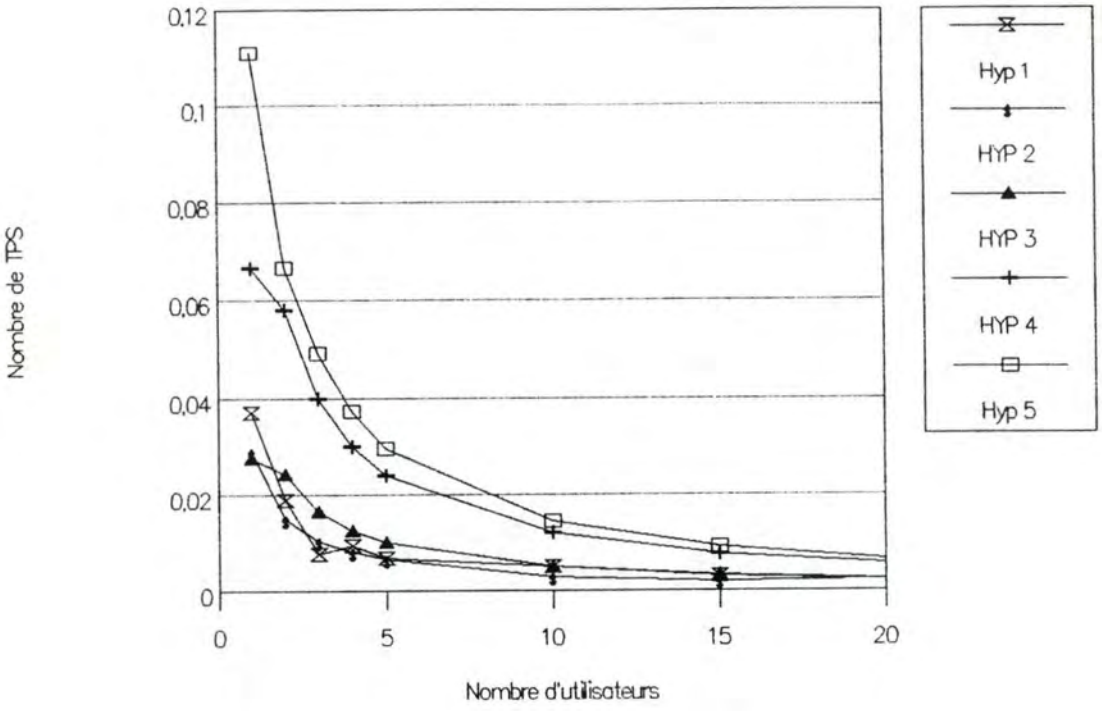


Figure 26 : ORACLE : Sélection simple

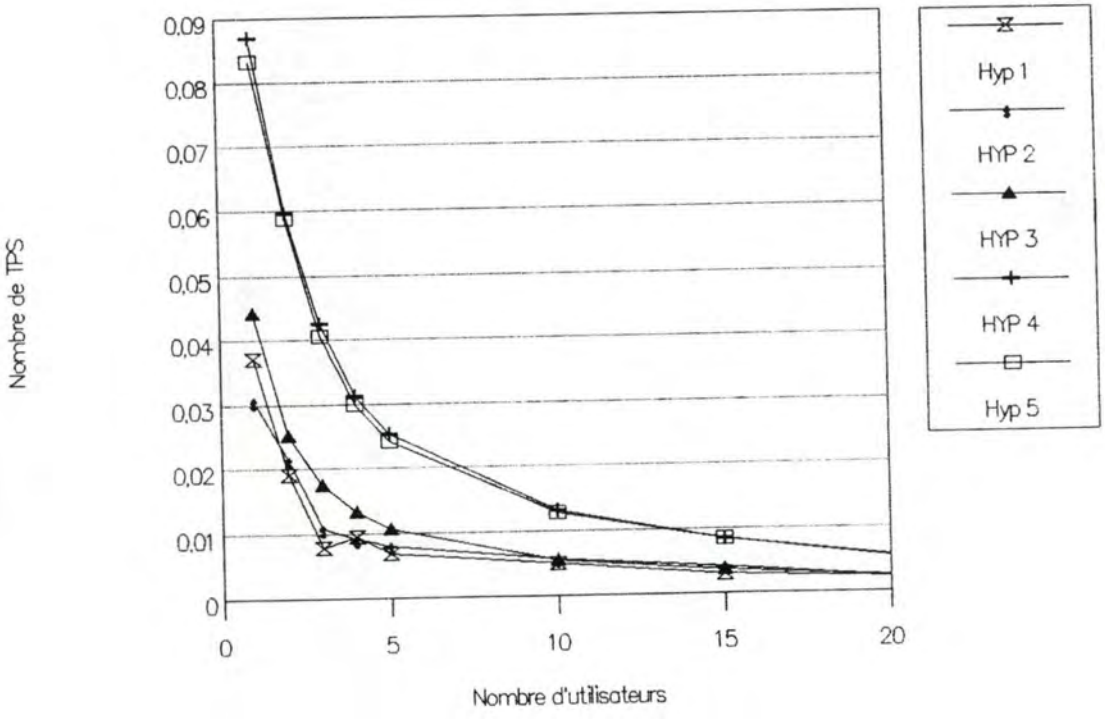


Figure 27 : ORACLE : Sélection complexe

6.2.1.3 Résultats par type d'hypothèse pour les différentes requêtes

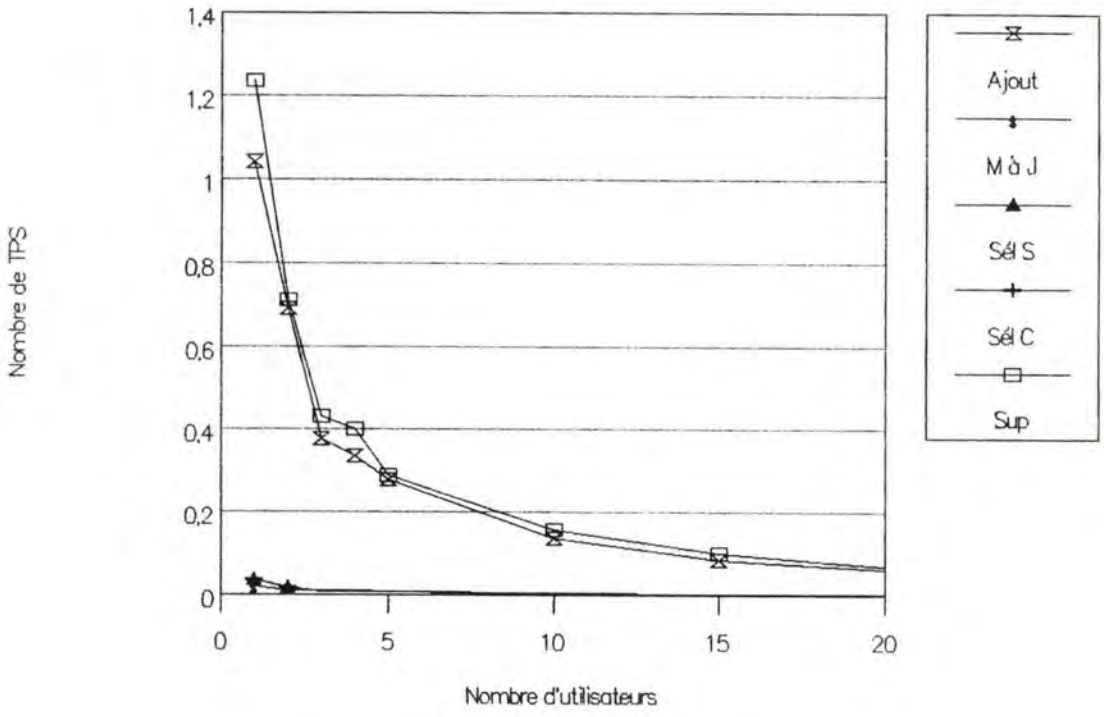


Figure 28 : ORACLE : Hypothèse 1

La figure 28 nous montre maintenant un autre genre de résultat. En effet, parmi toute les requêtes, l'absence d'index favorise nettement la création et la suppression d'enregistrements. Les autres requêtes sous cette hypothèse d'absence d'index ont un comportement très semblable et en nombre de TPS, elles sont inférieures à la création et à la suppression d'enregistrements.

Pour la deuxième hypothèse, nous remarquons sur la figure 29 que les requêtes donnant le plus de TPS sont à nouveau celles de création et suppression. Cependant, malgré une différence d'un peu moins de 50 % en nombre de TPS entre la mise à jour et la suppression dans le cas mono utilisateur, il faut noter que cette mise à jour est bonne par rapport aux deux autres requêtes de sélection.

L'évolution du nombre de TPS représentant le comportement des requêtes sous la troisième hypothèse nous donne un graphe semblable à celui de la figure précédente sous l'hypothèse 2.

Pour l'hypothèse 4, à la figure 30 comme cela à été précisé au paragraphe précédent, on ne retrouve pas les courbes d'évolution du nombre de TPS pour les requêtes d'ajout et de suppression d'enregistrements. Nous remarquons cependant que la mise à jour précède toujours les courbes de sélection.

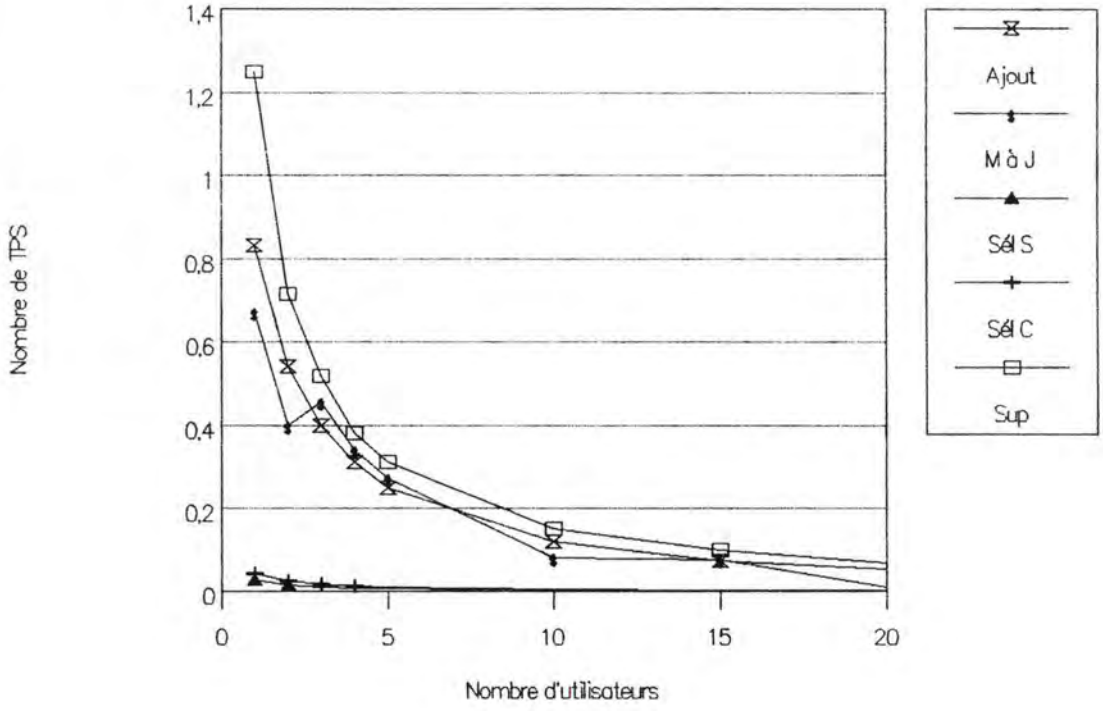


Figure 29 : ORACLE : Hypothèse 2

Et pour terminer, voici l'évolution du nombre de TPS des différentes requêtes pour l'hypothèse 5. Ainsi à la figure 31, nous voyons un certain bouleversement dans l'ordre des courbes puisque la mise à jour a dépassé la création et rejoint la suppression. Quant aux courbes de sélection, elles sont toujours dernières et assez loin derrière les trois premières; la taille de la base de données semble donc avoir une certaine influence.

6.2.1.4 Conclusions

De tout cela, nous pouvons tirer quelques conclusions. En effet, le principe du "clustering" des tables semble important et amener une grande amélioration dans les performances. Les index quant à eux n'apportent que

peu de résultats positifs. La moins bonne hypothèse est celle où nous avons modifié le nombre de tampons de données d'ORACLE. Ce résultat pourrait s'expliquer de la manière suivante : le nombre de tampons pour ORACLE augmente donc en mémoire et dès lors, le système a moins de place pour lui et il se voit forcé de faire plus de "swapping", ce qui ralentit les performances.

Nous pouvons aussi observer que parmi les cinq types de requêtes différentes, ce sont les sélections qui semblent les moins performantes. Cela peut aller à l'encontre d'idées reçues qui précisent qu'une base de données est faite pour la consultation. Une explication à ce phénomène pourrait être apportée par l'hypothèse suivante qui fait appel à la structure interne du RDBMS. En effet, lorsque l'on fait une demande à ORACLE pour une création, suppression ou mise à jour, il enregistre cette demande et la transmet à un autre processus qui exécutera le travail sur disque. Le premier processus ORACLE quant à lui a terminé son travail, et donc la requête demandée est, pour lui, terminée. Tandis que pour la sélection, le processus ORACLE ne transmet plus la demande à un autre, mais va chercher lui-même les enregistrements dans ses tables.

Quant à l'évolution en fonction du nombre d'utilisateurs, nous pouvons affirmer que l'on a une chute rapide des performances jusqu'à cinq utilisateurs, et que par la suite, une diminution beaucoup plus douce de celles-ci est observée.

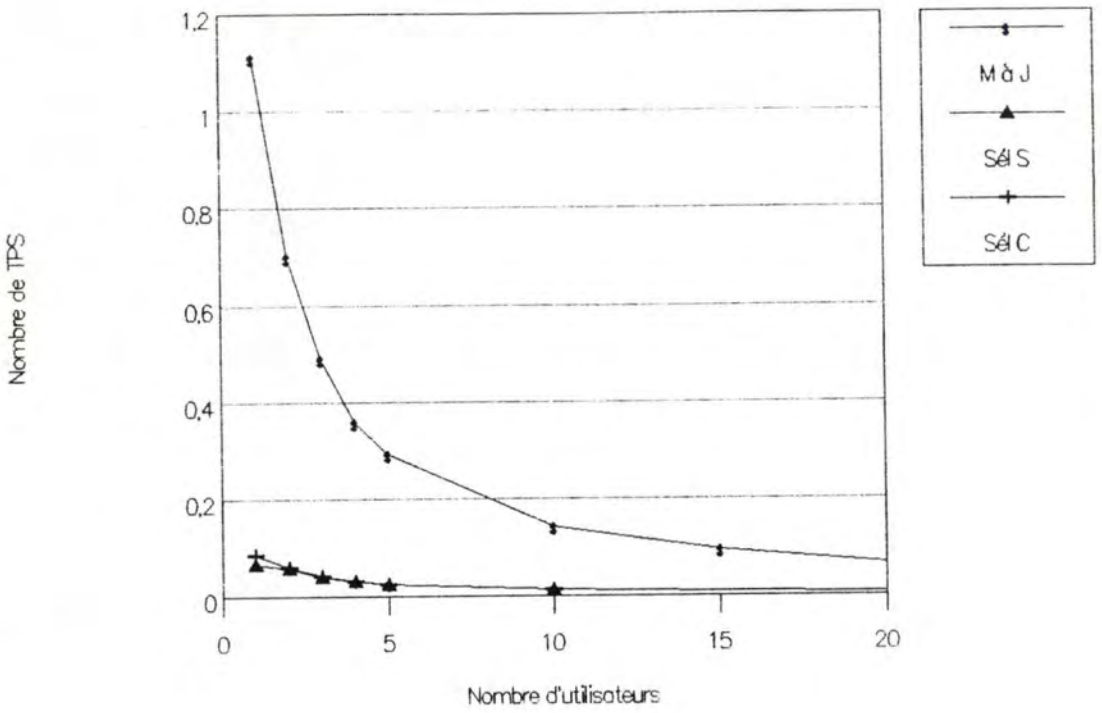


Figure 30 : ORACLE : Hypothèse 4

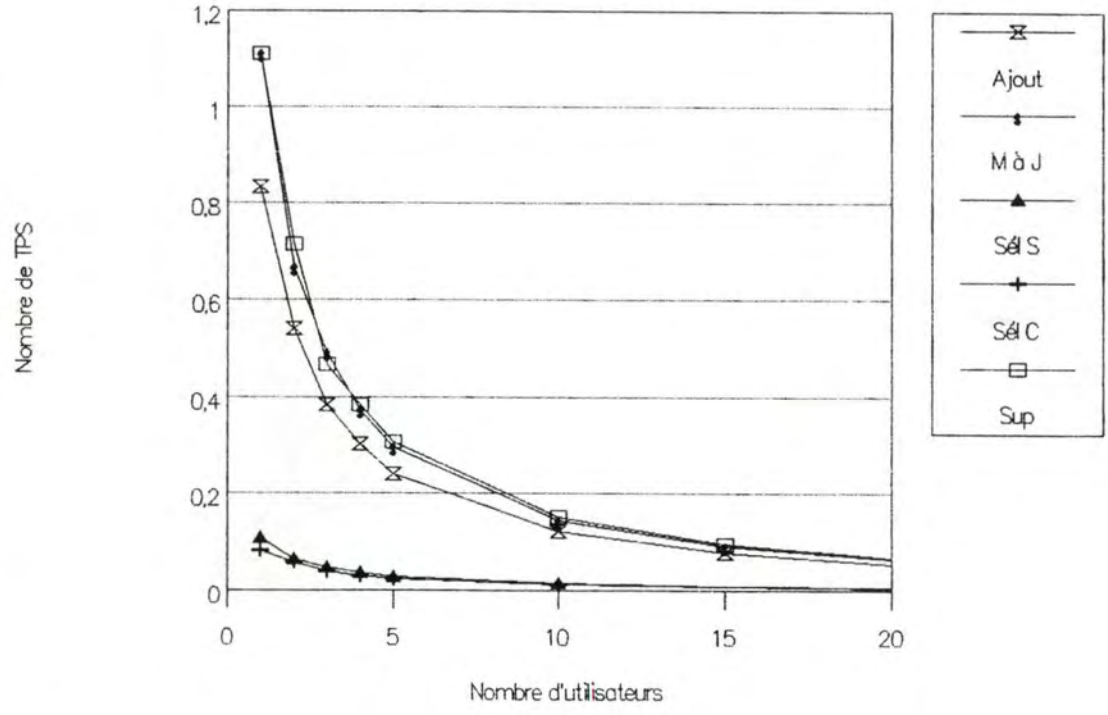


Figure 31 : ORACLE : Hypothèse 5

6.2.2 Pour INFORMIX

6.2ü2.1 Hypothèses

Pour INFORMIX, 5 hypothèses ont également été élaborées, mais étant donné que les possibilités des deux RDBMS ne sont pas identiques, quelques-unes des hypothèses présentées ci-dessous seront différentes par rapport au point précédent.

- **Hypothèse 1** : nous avons ici défini la même hypothèse que l'hypothèse 1 d'ORACLE;

- **Hypothèse 2** : dans ce cas, nous retrouvons la même hypothèse que l'hypothèse 2 d'ORACLE;

- **Hypothèse 3** : pour celle-ci, nous n'avons pas la possibilité de créer des tables "clustered", mais par contre, sur un seul index de chaque table, on peut créer un "clustering". Le "clustering" INFORMIX est totalement différent de celui d'ORACLE; en effet, ici on agit sur les index et lorsqu'ils sont "**clustered**", cela signifie que leurs enregistrements sont rangés physiquement en ordre croissant;

- **Hypothèse 4** : cette hypothèse est identique à la précédente sauf que la base de données aura une taille 2;

- **Hypothèse 5** : nous reprenons l'hypothèse 4, mais en modifiant un paramètre du système d'exploitation, à savoir la taille des tampons qui passent de 8 Mb à 2 Mb (cette taille étant quelque peu inférieure à celle de la base de données).

Avant d'aller plus avant dans l'analyse des résultats, il faut noter que suite à quelques problèmes concernant la suppression d'enregistrements, nous n'avons des résultats pour cette requête que pour la cinquième hypothèse.

6.2.2.2 Résultats par type de requête pour les différentes hypothèses

Pour INFORMIX, en ce qui concerne l'évolution du nombre de TPS pour la création d'enregistrements (figure 32), l'hypothèse 3 n'est pas présentée car le "clustering" index pose quelques problèmes pour la

création. Nous pouvons remarquer sur cette figure 32 que dans ce cas, les courbes se confondent.

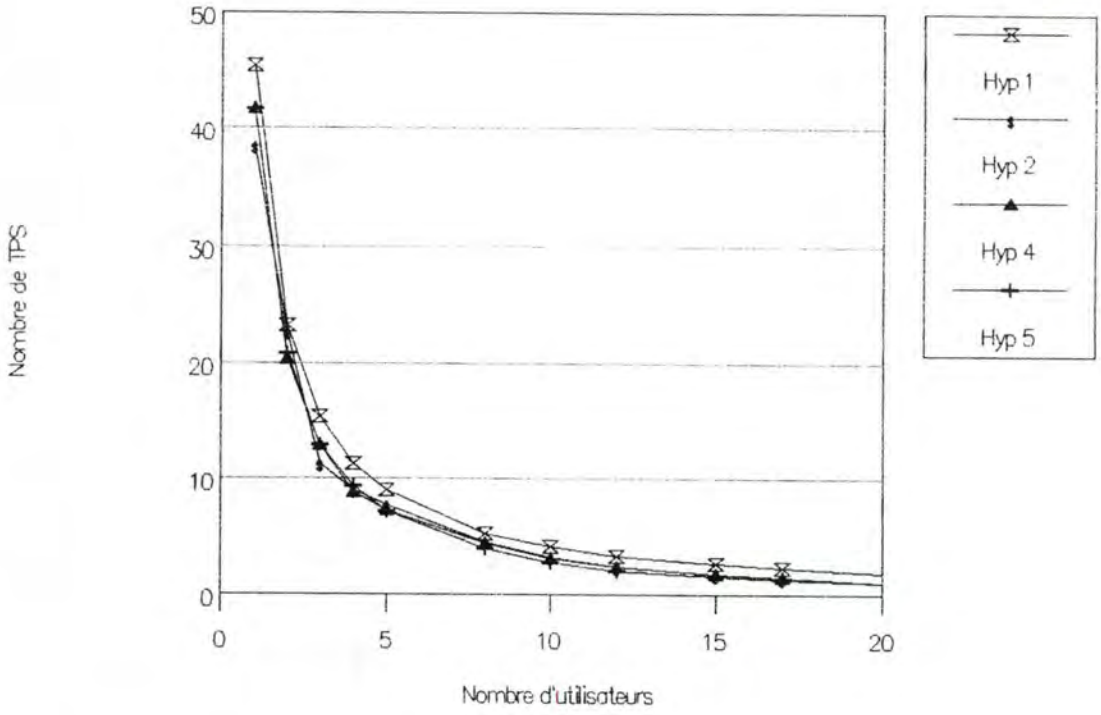


Figure 32 : INFORMIX : Ajout

A la figure 33 (évolution du nombre de TPS pour la mise à jour), nous observons à nouveau que toutes les courbes se confondent, à l'exception de celle qui correspond à l'hypothèse 1 c'est-à-dire la prise de mesure sans index. On ne voit d'ailleurs pas cette courbe tant elle se rapproche de l'axe des abscisses. La seule différence que nous distinguons parmi les autres courbes se situe au niveau mono utilisateur. Nous pouvons également noter que l'index "clustered" pour les bases de données de taille 2 semble amener un élément positif. En effet, les résultats semblent meilleurs sous cette hypothèse 4 que sous l'hypothèse 3. Cela paraît d'ailleurs très paradoxal.

En ce qui concerne l'évolution du nombre de TPS pour la sélection simple, toutes les courbes se confondent et aucune observation pertinente ne peut être donnée.

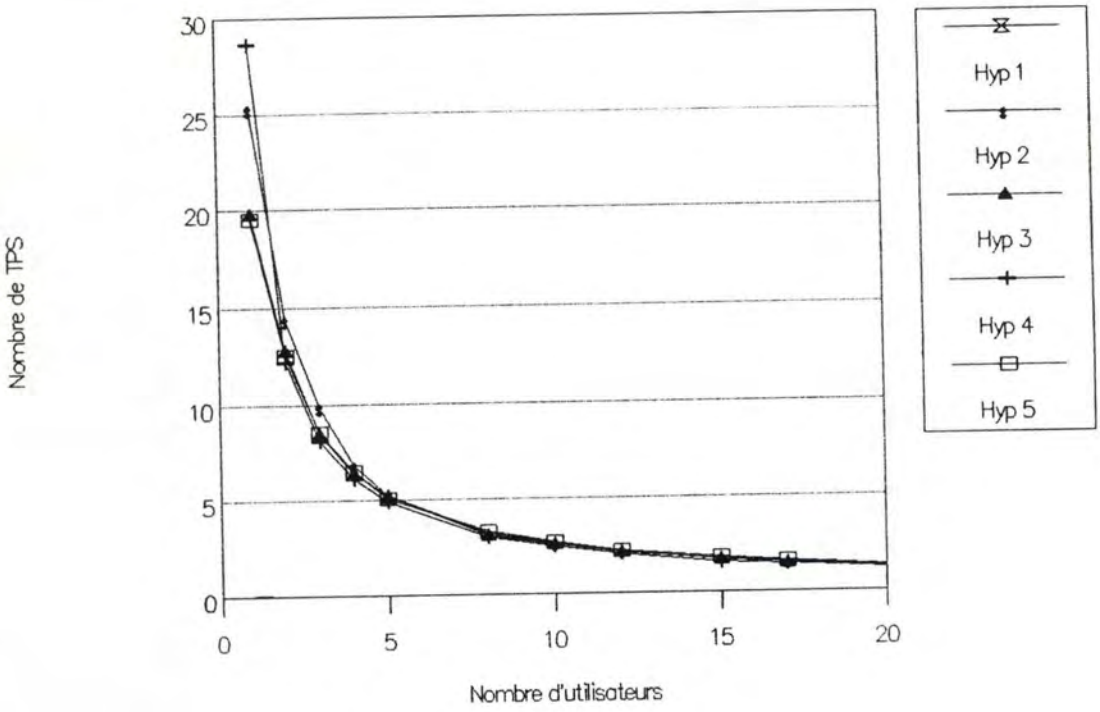


Figure 33 : INFORMIX : Mise à jour

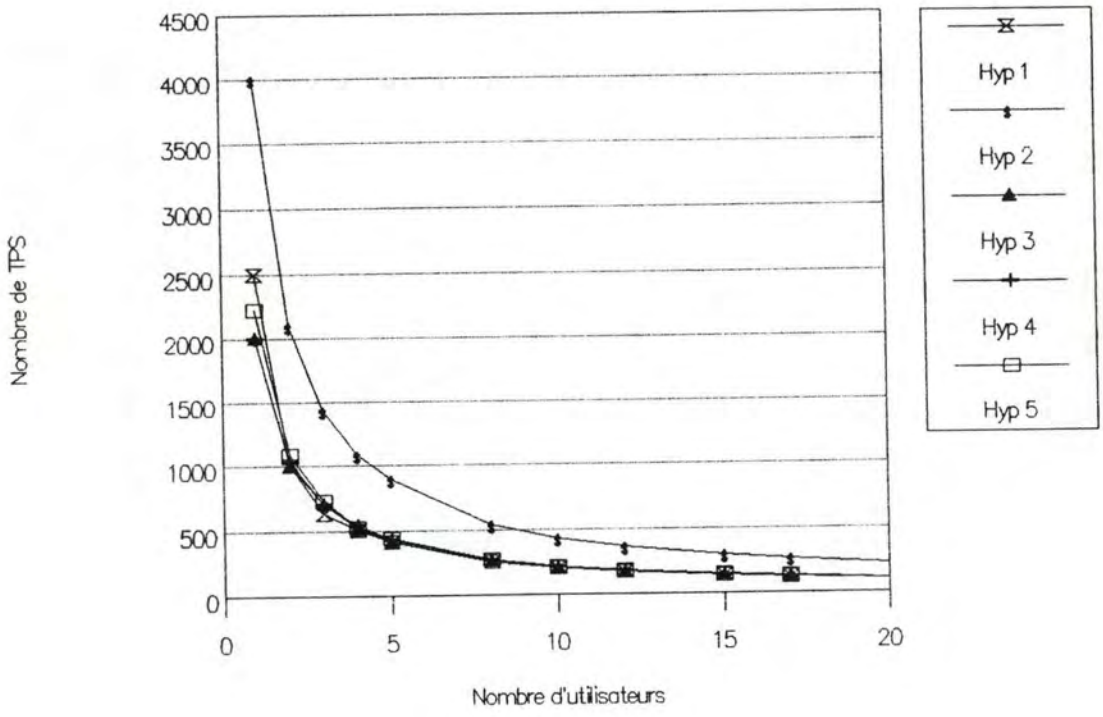


Figure 34 : INFORMIX : Sélection complexe

Le cas de l'évolution du nombre de TPS pour la sélection complexe sur le graphe de la figure 34, change quelque peu, puisque nous observons que la présence d'index (hypothèse 2) est très positive.

6.2.2.3 Résultats par type d'hypothèse pour les différentes requêtes

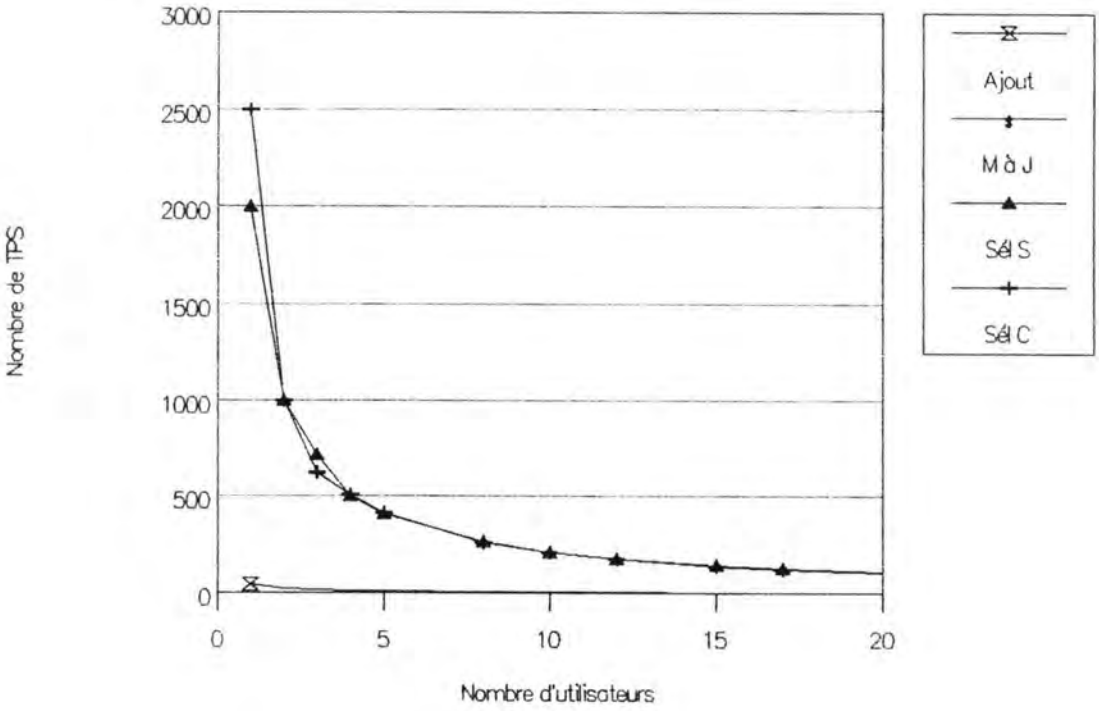


Figure 35 : INFORMIX : Hypothèse 1

Pour le cas de l'hypothèse 1, à la figure 35, nous remarquons que les sélections devancent nettement les autres requêtes en nombre de TPS (on ne peut d'ailleurs facilement les distinguer toutes), et ce avec une sélection complexe qui donne de meilleurs résultats que la sélection simple.

Sur la figure 36, nous retrouvons les mêmes conclusions que sur la précédente, avec encore plus renforcée cette différence encore inexpliquée entre la sélection simple et la sélection complexe.

L'hypothèse 3 que nous observons à la figure 37 nous fait à nouveau remarquer une grande différence entre les courbes de sélection et de mise à jour que nous pouvons à peine observer (la création d'enregistrement n'est pas présente pour cette hypothèse).

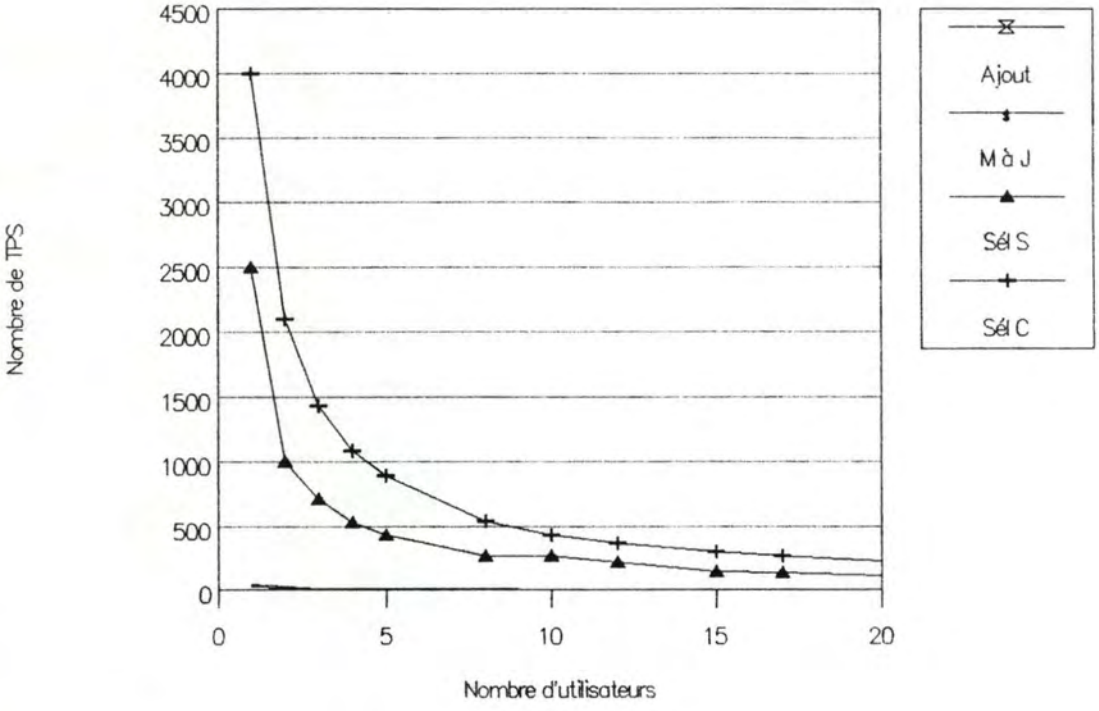


Figure 36 : INFORMIX : Hypothèse 2

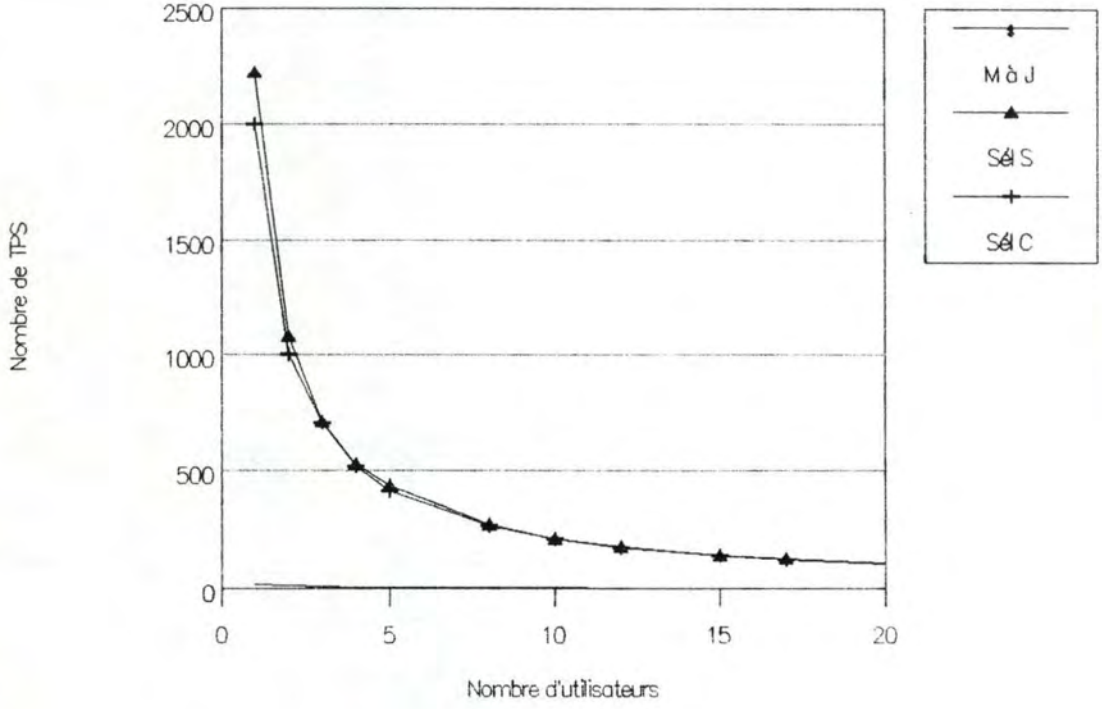


Figure 37 : INFORMIX : Hypothèse 3

L'hypothèse 4 et l'hypothèse 5 ne nous apportent aucune nouvelle information, nous obtenons des graphes d'allure semblable à celui de l'hypothèse 3 où la courbe montrant l'évolution du nombre de TPS pour l'ajout rejoint celle de la mise à jour.

6.2.2.4 Conclusions

Nous avons observé qu'INFORMIX semble très régulier. Seule la présence d'index amène quelques changements.

A l'opposé d'ORACLE, nous constatons que ce sont les courbes de sélection qui dominent en nombre de TPS, et ce sous toutes les hypothèses élaborées. La raison pour laquelle on observe un tel phénomène est dû à la structure même du RDBMS et pourrait être la suivante : le système de fichiers INFORMIX dans la version qui nous préoccupe suit une structure C-ISAM, ce qui lui permet d'aller rapidement rechercher les enregistrements qui lui sont demandés. En ce qui concerne la création ou mise à jour, INFORMIX, à l'opposé d'ORACLE, ne lègue pas l'information et le travail à un autre processus, et c'est seulement quand il a terminé qu'il rend la main à l'utilisateur.

Nous ajoutons de plus qu'en règle générale pour INFORMIX, le nombre de TPS chute très fort entre 1 et 2 utilisateurs. Ensuite, la pente des courbes devient plus douce et commence à stagner au niveau de 8 utilisateurs en moyenne.

Les observations suivantes que nous avons présentées, à savoir le fait que la sélection complexe donne de meilleurs résultats que la sélection simple, ou encore que la mise à jour dans une grande base de données donne de meilleurs résultats que la mise à jour effectuée dans une base de données de taille inférieure, restent encore sans hypothèses explicatives.

6.2.3 Pour ORACLE et INFORMIX

6.2.3.1 Hypothèses

Toujours pour cette raison d'hétérogénéité des deux RDBMS, nous avons essayé de réaliser la comparaison sur des hypothèses relativement comparables.

- **Hypothèse 1** : il s'agit ici de reprendre les résultats obtenus pour les deux hypothèses 1 des deux RDBMS, à savoir une base de données sans index;

- **Hypothèse 2** : celle-ci correspond à l'hypothèse 2 des deux RDBMS, c'est-à-dire une base de données avec index;

- **Hypothèse 3** : pour INFORMIX, nous reprenons l'hypothèse 3 (index "clustered") et pour ORACLE, l'hypothèse 4 (tables "clustered").

- **Hypothèse 4** : pour INFORMIX, nous reprenons l'hypothèse 4 (index "clustered" et base de données de taille 2) et pour ORACLE, l'hypothèse 5 (tables "clustered" et base de données de taille 2).

6.2.3.2 Comparaison des résultats par type de requête et par type d'hypothèse

Sur la figure 38, nous observons la variation du nombre de TPS en ce qui concerne l'ajout d'enregistrements sous la première hypothèse, c'est-à-dire qu'aucun index n'a été créé pour la base de données de taille 1. L'élément le plus frappant est bien évidemment le fait que le nombre de TPS pour ORACLE tourne toujours aux alentours de la valeur 0, tandis que pour INFORMIX, ce nombre commence à 45 dans un état mono utilisateur. Ces valeurs redescendent entre 5 et 2 pour 10 à 20 utilisateurs. INFORMIX donne donc un nombre de TPS nettement plus élevé.

L'évolution du nombre de TPS pour la création d'enregistrements sous les autres hypothèses, c'est-à-dire la deuxième et la troisième nous a montré des graphes d'allure absolument semblables.

Pour la mise à jour, nous remarquons à la figure 39 que, bien qu'ORACLE donne moins de TPS, la différence entre les deux RDBMS n'est pas très significative. Celle-ci se situe principalement quand il y a très peu d'utilisateurs en concurrence. A partir de 5 utilisateurs, nous pouvons dire que les deux courbes se confondent.

Lorsque nous ajoutons des index pour cette mise à jour (voir figure 40), ORACLE est immédiatement distancé par INFORMIX, cette différence diminuant lorsque le nombre d'utilisateurs augmente.

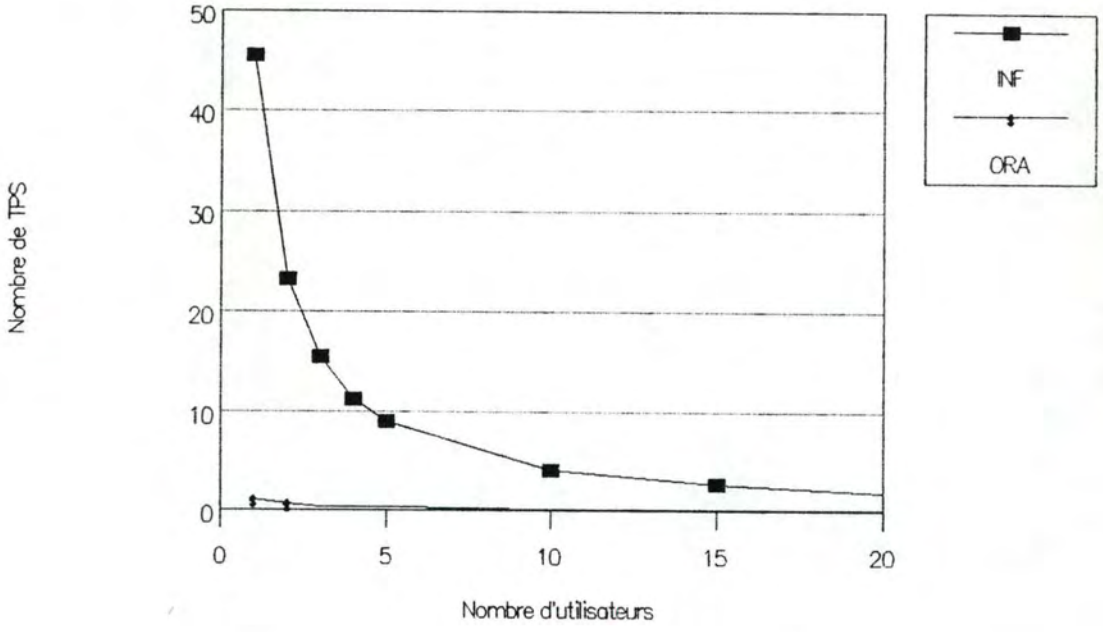


Figure 38 : INFORMIX - ORACLE : Ajout - Hypothèse 1

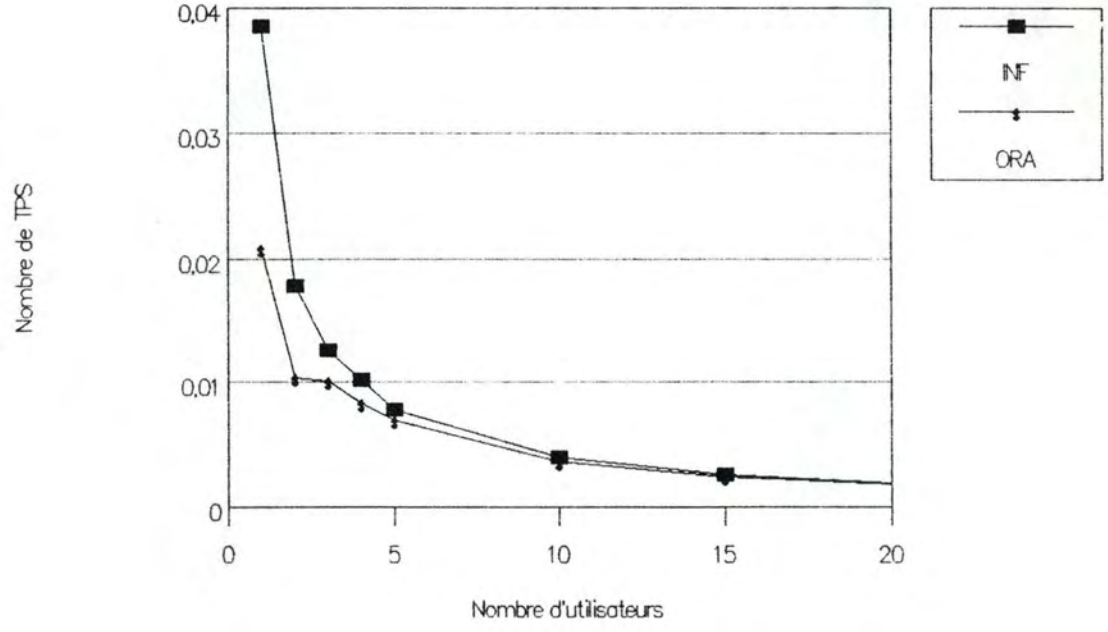


Figure 39 : INFORMIX - ORACLE : Mise à jour - Hypothèse 1

Sous les hypothèses 3 et 4, la différence entre ORACLE et INFORMIX reste semblable à celle observée à la figure 40 sous l'hypothèse 2. Il faut donc noter que le principe du "clustering" pour les tables d'ORACLE, bien que cela améliore ses propres performances, ne lui permet pas de rattraper INFORMIX qui lui est assez régulier dans ses performances.

Lorsque les sélections avaient été séparément étudiées pour ORACLE et INFORMIX, nous avions constaté que cette requête (simple ou complexe) était, parmi les requêtes analysées, la meilleure pour INFORMIX, et la moins bonne pour ORACLE. Dans ce cas, la différence qui avait été observée ci-dessus entre les deux RDBMS devrait donc encore s'accentuer pour les courbes de sélection. En effet, sur les graphes des figures 41 et 42, nous voyons l'évolution du nombre de TPS pour la sélection simple et la sélection complexe sous l'hypothèse 1. Dans ce cas, les résultats pour ORACLE sont tellement faibles par rapport à INFORMIX que la courbe du premier RDBMS se confond avec l'axe des abscisses.

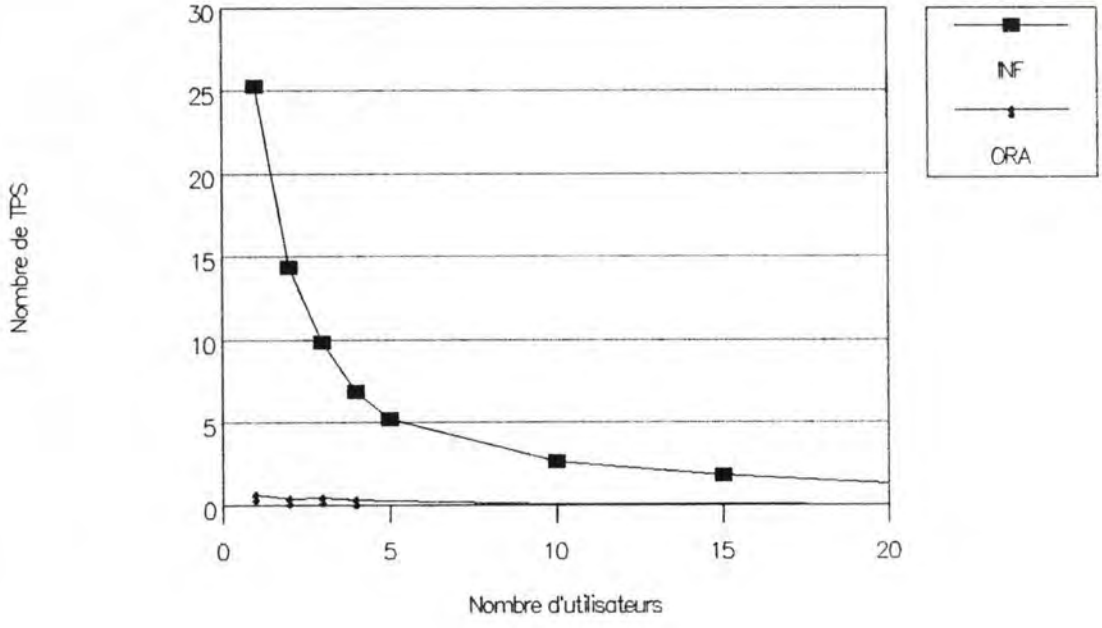


Figure 40 : INFORMIX - ORACLE : Mise à jour - Hypothèse 2

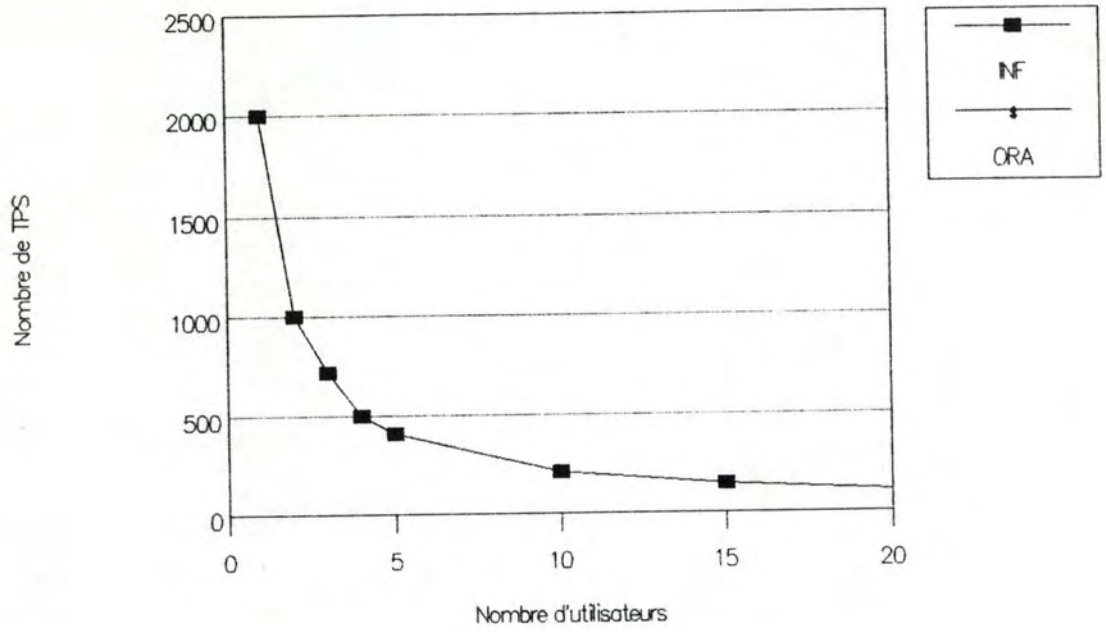


Figure 41 : INFORMIX - ORACLE : Sélection simple - Hypothèse 1

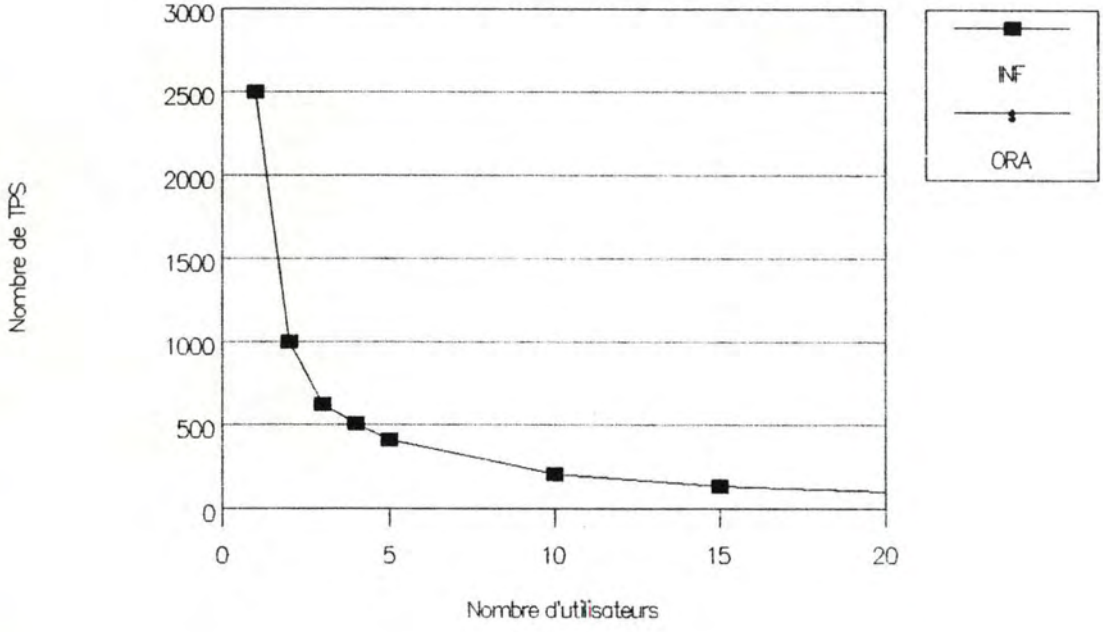


Figure 42 : INFORMIX - ORACLE : Sélection complexe - Hypothèse 1

En ce qui concerne les autres hypothèses pour l'évolution du nombre de TPS pour la sélection simple et complexe, le résultat obtenu est identique.

Pour la suppression, suite aux problèmes rencontrés avec INFORMIX, nous ne pouvons malheureusement pas donner de résultats pour cette partie de la comparaison.

6.2.3.3 Conclusions

Nous voyons donc qu'entre ORACLE et INFORMIX, il existe une énorme différence au niveau du nombre de TPS, et ce quelle que soit la requête. La seule hypothèse où les performances semblent se rapprocher est celle où il n'y a aucune création d'index; mais cela n'est valable que pour un seul type de requête : la mise à jour.

Cette différence entre ces deux RDBMS pourrait être expliquée par l'hypothèse suivante : le SQL d'ORACLE est interprété et le 4GL d'INFORMIX est compilé. Cette différence de conception aurait une influence prépondérante sur les performances. Dans un cas seulement cette différence n'est plus observée, cela pourrait s'expliquer par le fait que l'absence d'index pour la mise à jour en INFORMIX rend ce RDBMS très inefficace.

6.3 Analyse des résultats en fonction de la base de données réalisée

Bien que la base de données dont nous avons expliqué le développement au chapitre 5 ait été créée uniquement dans le but de ces tests, nous estimons être incomplet si une analyse des résultats obtenus ne se fait pas en fonction de cette base de données. En effet, nous ne pouvons nous arrêter à la théorie car ces RDBMS sont évidemment conçus pour supporter des applications bien réelles. Donc, en examinant cette base de données testée, et au vu des résultats théoriques, nous pouvons essayer de dégager le RDBMS le mieux adapté au problème.

Au chapitre 5, nous avons expliqué les diverses phases de la conception et de la création de la base de données. Plus précisément, au point 5.2.2, lors de la phase de conception logique, nous avons établi une liste des primitives pour l'utilisation de cette base de données, ainsi qu'une

quantification de ces dernières. Nous constatons que ce sont principalement des sélections qui sont demandées sur une journée, suivies des créations. Dès lors la première idée qui vient à l'esprit est d'utiliser, pour implémenter cette base de données, un RDBMS qui donne de très bons résultats pour les sélections et les mises à jours. Cependant, il ne faut pas perdre de vue l'évolution future de la base de données. Nous pourrions supposer qu'elle va non seulement augmenter de taille, mais que ses primitives vont également évoluer (création de nouvelles primitives de gestion de trésorerie par exemple). Avec tout ces éléments, nous pourrions envisager de modifier certains paramètres du RDBMS, ou certaines structures pour améliorer les performances.

Nous voyons donc que le choix du "bon" **RDBMS** n'est pas évident. Le tout est de bien savoir ce que l'on désire privilégier. D'un côté, on recherche un temps de réponse minimum pour l'utilisateur final (on pourrait alors choisir INFORMIX), et d'un autre, on estime que l'évolution future de la base de données sera telle que le choix d'ORACLE peut être envisagé.

6.4 Divers autres résultats et perspectives

6.4.1 Divers autres résultats

Nous pouvons, pour l'évaluation finale, également prendre en compte divers autres résultats.

Nous avons remarqué lors de ces tests, que lorsque l'on faisait augmenter régulièrement le nombre de requêtes pour une prise de temps, le nombre de TPS qui en résultait était constant. Cela peut amener à dire que le système est régulier.

Nous pouvons aussi noter que lorsque une requête ORACLE s'exécute, il faut se connecter à ce RDBMS avec vérification de nom d'utilisateur et mot de passe. La durée de cette connexion a été estimée à 4 secondes. Bien évidement, cela ralentit très fort les performances d'ORACLE, mais, cela ne suffit pas à combler la différence qui existe entre les deux RDBMS en matière de nombre de TPS.

Lorsque nous avons simulé plusieurs utilisateurs, une moyenne mathématique sur les temps d'exécution des requêtes avait été calculée afin de permettre d'établir le nombre de TPS. En même temps, l'écart-type mathématique a également été calculé. Ici encore, nous avons pu observer qu'il existait une grande différence entre ORACLE et INFORMIX. En effet, les écarts types sont beaucoup plus importants et irréguliers pour ORACLE (ceux-ci allant jusqu'à 40 % de la valeur de la moyenne) que pour INFORMIX. Cela nous amène encore à souligner la régularité que semble avoir INFORMIX.

Nous allons maintenant examiner aux figures 43 et (?) les temps de chargement (par les utilitaires **ODL** pour ORACLE et **DBLOAD** pour INFORMIX) des tables importantes que sont les tables MEMBRE et LICENCE, et ce sous différentes hypothèses.

	Table MEMBRE	Table LICENCE
- Base de données de taille 1	3 min. 13 sec.	2 min. 39 sec.
- Base de données de taille 1 avec tables clustered	11 min. 20 sec.	10 min. 19 sec.
- Base de données de taille 2 avec tables clustered	10 min. 32 sec.	10 min. 05 sec.

Figure 43 : Temps de chargement pour ORACLE

	Table MEMBRE	Table LICENCE
- Base de données de taille 1	2 min. 20 sec.	3 min. 02 sec.
- Base de données de taille 2	3 min. 02 sec.	5 min. 03 sec.

Figure 44 : Temps de chargement pour INFORMIX

Pour ORACLE, sur la figure 43, on peut constater que le temps de chargement de la table LICENCE est plus faible que celui de la table MEMBRE. Mais, bien que la table LICENCE contienne plus d'enregistrements que la table MEMBRE, ceux-ci sont plus petits et la taille résultante de la table LICENCE est en fait plus petite que celle de la table MEMBRE. Et donc cette différence s'explique aisément.

Quant au temps de chargement des tables quand elles sont "clustered", il est nettement supérieur, ce qui est tout à fait compréhensible étant donné le format de celles-ci.

Pour la différence de temps entre la base de données de taille 1 et celle de taille 2, elle n'est pas très significative.

Pour INFORMIX, à la figure 44, nous ne pouvons analyser les temps de chargement des tables que pour les deux tailles des bases de données. Dans ce cas, on ne retrouve pas ce qu'on avait observé pour ORACLE au niveau des tables MEMBRE et LICENCE. Cela pourrait s'expliquer logiquement par le fait que ORACLE travaille en longueur variable (ainsi que cela a été expliqué au point 3.2.1.2 du chapitre 3) et INFORMIX, lui, travaille en longueur fixe. Il en résulte pour INFORMIX, que la table LICENCE qui contient deux fois plus d'enregistrements que la table MEMBRE est plus importante (malgré le fait que ses enregistrements soient plus courts).

Pour ORACLE et INFORMIX, on ne sait ici que comparer les temps de chargement pour la base de données de taille 1, et nous remarquons qu'INFORMIX est plus rapide pour la table MEMBRE et plus lent pour la table LICENCE. Aucune conclusion ne peut donc être clairement établie.

Pour terminer ce paragraphe concernant quelques résultats divers, nous pouvons dire que lors de l'étude de ces deux RDBMS, INFORMIX nous a paru plus convivial et plus facile d'apprentissage qu'ORACLE, malgré la complexité de sa syntaxe.

6.4.2 Autres perspectives

Plusieurs éléments de poursuite de ce travail peuvent être donnés.

Tout d'abord, en ce qui concerne le contenu des requêtes, et principalement les sélections, il peut être renforcé par des fonctions de maximum, minimum, ou des opérateurs logiques, etc ...

Au niveau de la taille des requêtes, des transactions plus complexes pourraient être élaborées et ainsi permettre une analyse en fonction du contenu et du degré de complexité de ces transactions.

Enfin, un troisième point très important. Il concerne la création de tests complémentaires et l'utilisation d'outils tels que ODS pour ORACLE (voir chapitre 3, point 3.4.4) qui nous auraient permis une analyse encore plus fine et nous aurait peut-être fourni peut être fournir de nouvelles hypothèses nous permettant d'expliquer certains comportements.

6.5 Résumé et conclusion

Tout au long de ce chapitre, nous avons examiné les diverses requêtes et hypothèses élaborées pour les tests ainsi que les résultats leur correspondant. C'est ainsi que nous avons d'abord analysé pour ORACLE et INFORMIX séparément l'évolution du nombre de transactions par seconde (TPS), de 1 à 20 utilisateurs sous deux angles différents : par type de requête pour les différentes hypothèses et par type d'hypothèse pour toutes les requêtes. Ensuite, nous avons défini, afin de comparer ORACLE et INFORMIX, 4 hypothèses, et les résultats ont été analysés par requête et par hypothèse.

Tous les résultats que nous avons obtenus peuvent être récapitulés dans le tableau de la figure 45. Dans une première colonne, nous avons donné quelques critères établis en fonction des diverses conclusions de ce chapitre. Parmi ces critères, nous allons examiner si les deux RDBMS se valent, ou si l'un est meilleur que l'autre. Dans ce but, nous définissons, pour la deuxième colonne, les mêmes sigles que ceux de la conclusion traitant de la comparaison qualitative au point 3.6 du chapitre 3. Il s'agit donc :

- = : ORACLE et INFORMIX se valent
- o : ORACLE semble meilleur qu'INFORMIX (oo : bien meilleur, ...)
- i : INFORMIX semble meilleur qu'ORACLE (ii : bien meilleur, ...)

Nous voyons donc dans ce tableau de la figure 45 qu'INFORMIX distance quelque peu ORACLE.

Le dernier point abordé dans ce chapitre a été l'analyse des résultats obtenus en fonction de la base de données testée. Il ressort de cette analyse qu'il n'est pas évident de déterminer quel produit sera le mieux adapté pour l'application, bien que les avantages et inconvénients théoriques des performances soient établis.

Pour terminer ce chapitre, nous pouvons dire que toutes ces conclusions pourraient être affinées et complétées par d'autres genres de tests, les performances étant un domaine très ouvert où l'on peut toujours progresser.

- Requête de sélection	i
- Requête de mise à jour	i
- Requête d'ajout	o
- Requête de suppression	o
- Possibilité de modifier des paramètres du RDBMS avec effet positif	oo
- Pente des courbes d'évolution de TPS 1-2 utilisateurs (Hypothèses et requêtes confondues)	o
- Pente des courbes d'évolution de TPS 2-20 utilisateurs (Hypothèses et requêtes confondues)	i
- Nombre de TPS 1-20 utilisateurs	ii
- Régularité dans les résultats	ii
- Facilité d'utilisation	i
- Performance des utilitaires de chargement de données ...	=
- Présence de phénomènes inexpliqués	=

Figure 45 : Récapitulatif de la comparaison quantitative

7. Conclusion

Tout au long de ce travail, nous avons donc analysé d'un point de vue qualitatif et quantitatif les deux gestionnaires de base de données relationnelles que sont ORACLE (de ORACLE Corporation) et INFORMIX (de INFORMIX Software INC.).

Avant toute chose, nous avons présenté le contexte d'étude, c'est-à-dire que nous avons précisé quels seraient les éléments et les versions des deux RDBMS qui allaient être étudiés. Ensuite, nous avons présenté la machine sur laquelle s'est déroulé l'ensemble des tests pour la comparaison quantitative.

Nous avons ensuite réalisé une comparaison théorique pour établir la comparaison qualitative. Il en est ressorti, que pour des points importants comme la protection ou la mise au point pour de meilleures performances, ORACLE domine nettement INFORMIX. Par contre, au niveau de l'accès pour les utilisateurs, c'est INFORMIX qui devance ORACLE.

La phase suivante a été la préparation de la comparaison qualitative. En effet, au chapitre 4, nous avons donné quelques éléments théoriques sur la conception des bases de données ainsi que sur les mesures de performances. Pour passer ensuite au chapitre 5 où nous avons développé la conception et la création d'une base de données dans le but de réaliser, au chapitre 6, des mesures de performances.

Nous arrivons donc à ce chapitre 6 relatif à la comparaison qualitative entre ORACLE et INFORMIX. D'un point de vue tout à fait théorique, le résultat des tests amène à affirmer qu'INFORMIX, en rapidité, est le meilleur et il permet de retrouver rapidement des éléments de la base de données. De plus, il offre une très grande régularité.

Nous voyons donc que au niveau de la comparaison qualitative, ORACLE prend un avantage sur INFORMIX, et du point de vue quantitatif, c'est le contraire.

Mais à cela, il faut ajouter deux points très importants :

1. Ainsi que nous l'avons vu au paragraphe 6.4 du chapitre 6, les résultats théoriques de la comparaison quantitative doivent être analysés en fonction de l'application que l'on désire développer. Et il n'est pas évident de dire quel sera le RDBMS le mieux adapté à l'application. Mais en plus, si on ajoute les résultats de la comparaison qualitative, le choix devient encore plus difficile et dépend principalement des hypothèses que l'on émet sur l'évolution future de l'application.

2. Le deuxième point important concerne le développement de l'application. Il s'agit du degré de facilité (ou de difficulté) que l'on accepte pour développer l'application. En effet, à ce niveau, en étudiant ces deux gestionnaires, il nous a semblé qu'INFORMIX-4GL était nettement plus convivial et simple d'utilisation (malgré la complexité de certaines de ses structures) qu'ORACLE; et donc, le développement d'une application avec INFORMIX pourrait être plus rapide qu'avec ORACLE.

En conclusion, il nous semble que ces deux gestionnaires de base de données que sont ORACLE et INFORMIX se valent, on ne peut affirmer que l'un est meilleur que l'autre. Cependant, nous pouvons dire que, en fonction de l'application, selon que l'on désire la développer avec plus ou moins de facilités, plus ou moins rapidement, et suivant son évolution future, un produit est mieux adapté que l'autre. Ainsi, dans le cas qui nous occupe, si nous considérons que l'application va assez bien évoluer dans le futur, et que les problèmes de sécurité, de classe d'utilisateurs sont plus importants que le temps de réponse lorsque ces derniers font une requête, alors ORACLE est mieux adapté. Par contre, si le temps de réponse pour une transaction est le point estimé le plus important, et que l'application dans le futur n'évoluera pas beaucoup, INFORMIX nous semble mieux convenir.

Nous pouvons donc constater à la fin de ce travail que, bien que nous nous trouvons face à deux gestionnaires de bases de données relationnelles, ceux-ci ont une conception et une finalité totalement différentes. Faire un choix entre ces deux produits n'est pas chose aisée.

Bibliographie

[BACH,86]
Maurice J. BACH
The Design of the UNIX Operating System
Prentice/Hall International INC., 1986.

[BIJI,sd]
Albert BIJI & Ralph BURROUGH
Onderzoek naar de Eigenschappen van RDBMSD en onder UNIX
Hogeschool Oost Nederland, NTS "de Meare", afdelingHIOte Enschede.

[BITTO,83]
Dina BITTON, David J. DeWITT & Carolyn TURBYFILL
Benchmarking Database Systems, a Systematic Approach
in : *Ninth International conference on Very Large Data Bases Proceedings*, Florence, Italy, October 31 November 2, 1983.

[BODAR,88]
François BODART
Conception des systèmes d'informations
Cours dispensé aux F.U.N.D.P., NAMUR, 1987/1988.

[BOURN,85]
Steve BOURNE
Le système UNIX
Interédition, Paris, 1985.

[DATAD,88]
DATA DECISION 84
Drie vragen, drie antwoorden : RDBMS en SQL
Data Decision 84, 31 Augustus 1988.

[DATE,86]
C.J. DATE
Relationnal Database, Selected Writings
Addison-Wesley Publishing Company, USA, 1986.

[DONKI,sd]
Richard DONKIN
UNISYS 4GL Seminar. 4GL and Database Technology for UNIX System, the Intrduction Set.

[FERRA,83]
Domenico FERRARI, Guiseppe SERRAZZI & Alessandro ZEIGNER
Measurement and Tuning og Computer Systems
Prentice Hall, INC., USA, 1983.

[HAINA,86]
J.-L. HAINAUT
Conception assistée des applications informatiques : conception de la base de données
Masson, Presse universitaires de Namur, 1986.

[HAINA,90]
J.-L. HAINAUT
Technologie des bases de données, matières approfondies
Cours dispensé aux F.U.N.D.P., NAMUR, 1990.

[HANSO,88]
Eric N. HANSON
Processing Queries Against Database Procedures : A Performance Analysis
in : *SIGMOD International Conference on Management of Data*, ACM Press, New York, 1988.

[ISI1,87]
INFORMIX Software INC.
INFORMIX 4GL, SQL Based application Development Language, User guide
INFORMIX Software INC., June 1987.

[ISI2,87]
INFORMIX Software INC.
INFORMIX 4GL, SQL Based Application Development Language, Reference Manual
INFORMIX Software INC., vol. 1, June 1987.

[ISI3,87]
INFORMIX Software INC.
INFORMIX 4GL, SQL Based Application Development Language, Reference Manual
INFORMIX Software INC., vol.2, June 1987.

[ISI4,89]
INFORMIX Software INC.
How to Benchmark and Tune an OLTP Application
INFORMIX Software INC., 1989.

[KERNI,86]
Brian W. KERNIGHAN et Dennis M. RITCHIE
Le Langage C
Masson, Paris, 1986

[NIERE,86]
Nicolas Nierenberg
Rules for Benchmarking DBMS
UNIFY Corporation, USA, 1986.

[ORAC1,86]
ORACLE Corporation
The ORACLE Database Administrator's Guide (version 5.1)
ORACLE Corporation, n°3601, August 19th 1986.

[ORAC1,87]
ORACLE Corporation
*SQL*Forms Release Notes (version 2.0)*
ORACLE Corporation, n°3004, January 14th 1987.

[ORAC2,86]
ORACLE Corporation
RDBMS Utilities User's Guide (version 5.1)
ORACLE Corporation, n°3602, July 21st 1986.

[ORAC3,86]
ORACLE Corporation
ORACLE RDBMS Release Notes (version 5.1)
ORACLE Corporation, n°3001, 1986.

[ORAC4,86]
ORACLE Corporation
*SQL*PLUS User's guide (version 2.0)*
ORACLE Corporation, n°3201, 1986.

[ORAC5,86]
ORACLE Corporation
*SQL*PLUS Reference Guide (version 2.0)*
ORACLE Corporation, n°3203, 1986.

[ORAC6,86]
ORACLE Corporation
*SQL*PLUS Release Notes (version 2.0)*
ORACLE Corporation, n°3003, 1986.

[ORAC7,86]
ORACLE Corporation
*SQL*Forms Designer's Tutorial (version 2.0)*
ORACLE Corporation, n°3302, 1986.

[ORAC8,86]
ORACLE Corporation
*SQL*Forms Operator's Guide (version 2.0)*
ORACLE Corporation, n°3301, 1986.

[ORAC9,86]
ORACLE Corporation
*SQL*REPORT User's Guide (version 1.0)*
ORACLE Corporation, n°3603, July 3rd 1986.

[OXFOR,83]
OXFORD University Press
Dictionnary of Computing
OXFORD University Press, New York, 1983

[RAMAE,89]
Jean RAMAEKERS
Le choix d'un système informatique
Cours F.U.N.D.P. Namur, mars 1989.

[RODOF,89]
David RODOFF
4GL Unbundled
UNIX World, June 1989.

[SCHLU,89]
Louis SCHLUETER
4th Generztion Languages : Breaking the Programming Bottleneck
Unisphere, April 1989.

[SKRIN,87]
Richard SKRINDE
Evaluating UNIX DBMS Products
UNIX World, February 1987.

[SOBEL,84]
Mark G. SOBELL
A Practical Guide to the UNIX System
The Benjamin/Cummings Publishing Company, INC., 1984.

[SOFTO,89]
Softools Systems
INFORMIX DBA, The DBA Workshop
Softools Systems, 1989.

[STAMP,89]
David STAMPS
Case vs 4GLs
Datamation, August 15 1989.

[TANDE,88]
The Tandem Processing Group
A Benchmark of NonStop SQL on Debit Credit Transaction
in : *SIGMOD International Conference on Management of Data*, ACM Press, New York, 1988.

[UNIS1,88]
UNISYS Corporation
ORACLE ™ (version 5) SQL Workshop, Student Guide
UNISYS Corporation, September 1988.

[UNIS1,89]
UNISYS Corporation
U 6000/50, U Series Processor
UNISYS Corporation, January 1989.

[UNIS2,88]
UNISYS Corporation
*ORACLE ™ (version 5) SQL*FORMS Workshop, Student Guide*
UNISYS Corporation, September 1988.

[UNIS2,89]
UNISYS Corporation
U Series Data Management INFORMIX RDBMS
UNISYS Corporation, September 1989.

[UNIS3,88]
UNISYS Corporation
ORACLE ™ (version 5) Database Administration, Student Guide
UNISYS Corporation, September 1988.

[UNIS3,89]
UNISYS Corporation
U Series Data Management ORACLE RDBMS
UNISYS Corporation, September 1989.

[UNIS4,89]
UNISYS Corporation
U 6000 Series Your Productivity Platform
UNISYS Corporation, 1989.

[WISTO,88]
Alan WINSTON
What's different about the many fourth-generation languages on the market today ? And how do you choose about them ?
UNIX World, September 1988.

[YAO,87]
S. Bing YAO, Alan R. HEVNER & Hélène YOUNG-MYERS
Analysis of Database System Architecture Using Benchmarks
in : *IEEE Transactions on software engineering*, vol.SE 13, n°6, June 1987.

Index

Annexe 1

Codes associés à l'élaboration et au remplissage automatisé de la base de données

```
#
# ORACLE   -   FICHIER DE GENERATION DE LA BASE DE DONNEES
#
# ORACLE is a registred trademark of ORACLE CORPORATION
#

clear
echo ""
echo "Ce script va creer la Database -sport-, ses tables, et"
echo "ses records."
echo "**************************************************"
echo ""
echo "ENTER quand vous voulez ..."
read ATTENTE

clear
echo "Creation de la database et de tables ..."
echo "Temps requis pour la creation de la Database et des
tables" > timeo
echo
"**************************************************"
>> timeo
time 2>>timeo sqlplus -silent system/manager @cdbs.sql > bidon

echo "\
1    CSP_MULTI_SPORTS     RUE VANDAMME  405000          NAMUR
32-81-3310555\
" > cspo.asc

echo "\
 1              CB_JUDO     RUE VANDAMME  405000          NAMUR
32-81-33105561
 2            CB_AIKIDO     RUE VANDAMME  405000          NAMUR
32-81-33105571
 3            CB_KARATE     RUE VANDAMME  405000          NAMUR
32-81-33105581
 4               CB_TIR     RUE VANDAMME  405000          NAMUR
32-81-33105591
 5           CB_TIR_ARC     RUE VANDAMME  405000          NAMUR
32-81-33105601
 6         CB_ESCALADE     RUE VANDAMME  405000          NAMUR
32-81-33105611
 7         CB_FOOTBALL     RUE VANDAMME  405000          NAMUR
32-81-33105621
 8       CB_VOLLEYBALL     RUE VANDAMME  405000          NAMUR
32-81-33105631
 9         CB_HANDBALL     RUE VANDAMME  405000          NAMUR
32-81-33105641
10        CB_BADMINTON     RUE VANDAMME  405000          NAMUR
32-81-33105651
11         CB_NATATION     RUE VANDAMME  405000          NAMUR
32-81-33105661
12           CB_SQUASH     RUE VANDAMME  405000          NAMUR
32-81-33105671
13           CB_TENNIS     RUE VANDAMME  405000          NAMUR
32-81-33105681
14      CB_GYMNASTIQUE     RUE VANDAMME  405000          NAMUR
32-81-33105691
```

```
15        CB_ATHLETISME     RUE VANDAMME  405000          NAMUR
32-81-33105701\
" > cbo.asc

echo "\
 1          CBA_JUDO_1     RUE ADVERSE  1015000          NAMUR
32-81-739056
 2          CBA_JUDO_2     RUE ADVERSE  1025000          NAMUR
32-81-739057
 3          CBA_JUDO_3     RUE ADVERSE  1035000          NAMUR
32-81-739058
 4          CBA_JUDO_4     RUE ADVERSE  1045000          NAMUR
32-81-739059
 5          CBA_JUDO_5     RUE ADVERSE  1055000          NAMUR
32-81-739060
 6          CBA_JUDO_6     RUE ADVERSE  1065000          NAMUR
32-81-739061
 7        CBA_AIKIDO_1     RUE ADVERSE  1075000          NAMUR
32-81-739062
 8        CBA_AIKIDO_2     RUE ADVERSE  1085000          NAMUR
32-81-739063
 9        CBA_AIKIDO_3     RUE ADVERSE  1095000          NAMUR
32-81-739064
10        CBA_AIKIDO_4     RUE ADVERSE  1105000          NAMUR
32-81-739065
11        CBA_AIKIDO_5     RUE ADVERSE  1115000          NAMUR
32-81-739066
12        CBA_AIKIDO_6     RUE ADVERSE  1125000          NAMUR
32-81-739067
13           CBA_TIR_1     RUE ADVERSE  1135000          NAMUR
32-81-739068
14           CBA_TIR_2     RUE ADVERSE  1145000          NAMUR
32-81-739069
15           CBA_TIR_3     RUE ADVERSE  1155000          NAMUR
32-81-739070
16           CBA_TIR_4     RUE ADVERSE  1165000          NAMUR
32-81-739071
17           CBA_TIR_5     RUE ADVERSE  1175000          NAMUR
32-81-739072
18           CBA_TIR_6     RUE ADVERSE  1185000          NAMUR
32-81-739073
19       CBA_TIR_ARC_1     RUE ADVERSE  1195000          NAMUR
32-81-739074
20       CBA_TIR_ARC_2     RUE ADVERSE  1205000          NAMUR
32-81-739075
21       CBA_TIR_ARC_3     RUE ADVERSE  1215000          NAMUR
32-81-739076
22       CBA_TIR_ARC_4     RUE ADVERSE  1225000          NAMUR
32-81-739077
23       CBA_TIR_ARC_5     RUE ADVERSE  1235000          NAMUR
32-81-739078
24       CBA_TIR_ARC_6     RUE ADVERSE  1245000          NAMUR
32-81-739079
25      CBA_ESCALADE_1     RUE ADVERSE  1255000          NAMUR
32-81-739080
26      CBA_ESCALADE_2     RUE ADVERSE  1265000          NAMUR
32-81-739081
```

27	CBA_ESCALADE_3	RUE ADVERSE	1275000	NAMUR	32-81-739082
28	CBA_ESCALADE_4	RUE ADVERSE	1285000	NAMUR	32-81-739083
29	CBA_ESCALADE_5	RUE ADVERSE	1295000	NAMUR	32-81-739084
30	CBA_ESCALADE_6	RUE ADVERSE	1305000	NAMUR	32-81-739085
31	CBA_FOOTBALL_1	RUE ADVERSE	1315000	NAMUR	32-81-739086
32	CBA_FOOTBALL_2	RUE ADVERSE	1325000	NAMUR	32-81-739087
33	CBA_FOOTBALL_3	RUE ADVERSE	1335000	NAMUR	32-81-739088
34	CBA_FOOTBALL_4	RUE ADVERSE	1345000	NAMUR	32-81-739089
35	CBA_FOOTBALL_5	RUE ADVERSE	1355000	NAMUR	32-81-739090
36	CBA_FOOTBALL_6	RUE ADVERSE	1365000	NAMUR	32-81-739091
37	CBA_VOLLEYBALL_1	RUE ADVERSE	1375000	NAMUR	32-81-739092
38	CBA_VOLLEYBALL_2	RUE ADVERSE	1385000	NAMUR	32-81-739093
39	CBA_VOLLEYBALL_3	RUE ADVERSE	1395000	NAMUR	32-81-739094
40	CBA_VOLLEYBALL_4	RUE ADVERSE	1405000	NAMUR	32-81-739095
41	CBA_VOLLEYBALL_5	RUE ADVERSE	1415000	NAMUR	32-81-739096
42	CBA_VOLLEYBALL_6	RUE ADVERSE	1425000	NAMUR	32-81-739096
43	CBA_HANDBALL_1	RUE ADVERSE	1435000	NAMUR	32-81-739098
44	CBA_HANDBALL_2	RUE ADVERSE	1445000	NAMUR	32-81-739099
45	CBA_HANDBALL_3	RUE ADVERSE	1455000	NAMUR	32-81-739100
46	CBA_HANDBALL_4	RUE ADVERSE	1465000	NAMUR	32-81-739101
47	CBA_HANDBALL_5	RUE ADVERSE	1475000	NAMUR	32-81-739102
48	CBA_HANDBALL_6	RUE ADVERSE	1485000	NAMUR	32-81-739103
49	CBA_BADMINTON_1	RUE ADVERSE	1495000	NAMUR	32-81-739104
50	CBA_BADMINTON_2	RUE ADVERSE	1505000	NAMUR	32-81-739105
51	CBA_BADMINTON_3	RUE ADVERSE	1515000	NAMUR	32-81-739106
52	CBA_BADMINTON_4	RUE ADVERSE	1525000	NAMUR	32-81-739107
53	CBA_BADMINTON_5	RUE ADVERSE	1535000	NAMUR	32-81-739108
54	CBA_BADMINTON_6	RUE ADVERSE	1545000	NAMUR	32-81-739109
55	CBA_NATATION_1	RUE ADVERSE	1555000	NAMUR	32-81-739110

56	CBA_NATATION_2	RUE ADVERSE	1565000	NAMUR	32-81-739112
57	CBA_NATATION_3	RUE ADVERSE	1575000	NAMUR	32-81-739113
58	CBA_NATATION_4	RUE ADVERSE	1585000	NAMUR	32-81-739114
59	CBA_NATATION_5	RUE ADVERSE	1595000	NAMUR	32-81-739115
60	CBA_NATATION_6	RUE ADVERSE	1605000	NAMUR	32-81-739116
61	CBA_SQUASH_1	RUE ADVERSE	1615000	NAMUR	32-81-739117
62	CBA_SQUASH_2	RUE ADVERSE	1625000	NAMUR	32-81-739118
63	CBA_SQUASH_3	RUE ADVERSE	1635000	NAMUR	32-81-739119
64	CBA_SQUASH_4	RUE ADVERSE	1645000	NAMUR	32-81-739120
65	CBA_SQUASH_5	RUE ADVERSE	1655000	NAMUR	32-81-739121
66	CBA_SQUASH_6	RUE ADVERSE	1665000	NAMUR	32-81-739122
67	CBA_TENNIS_1	RUE ADVERSE	1675000	NAMUR	32-81-739123
68	CBA_TENNIS_2	RUE ADVERSE	1685000	NAMUR	32-81-739124
69	CBA_TENNIS_3	RUE ADVERSE	1695000	NAMUR	32-81-739125
70	CBA_TENNIS_4	RUE ADVERSE	1705000	NAMUR	32-81-739126
71	CBA_TENNIS_5	RUE ADVERSE	1715000	NAMUR	32-81-739127
72	CBA_TENNIS_6	RUE ADVERSE	1725000	NAMUR	32-81-739128
73	CBA_GYMNASTIQUE_1	RUE ADVERSE	1735000	NAMUR	32-81-739129
74	CBA_GYMNASTIQUE_2	RUE ADVERSE	1745000	NAMUR	32-81-739130
75	CBA_GYMNASTIQUE_3	RUE ADVERSE	1755000	NAMUR	32-81-739131
76	CBA_GYMNASTIQUE_4	RUE ADVERSE	1765000	NAMUR	32-81-739132
77	CBA_GYMNASTIQUE_5	RUE ADVERSE	1775000	NAMUR	32-81-739133
78	CBA_GYMNASTIQUE_6	RUE ADVERSE	1785000	NAMUR	32-81-739134
79	CBA_ATHLETISME_1	RUE ADVERSE	1795000	NAMUR	32-81-739135
80	CBA_ATHLETISME_2	RUE ADVERSE	1805000	NAMUR	32-81-739136
81	CBA_ATHLETISME_3	RUE ADVERSE	1815000	NAMUR	32-81-739137
82	CBA_ATHLETISME_4	RUE ADVERSE	1825000	NAMUR	32-81-739138
83	CBA_ATHLETISME_5	RUE ADVERSE	1835000	NAMUR	32-81-739139
84	CBA_ATHLETISME_6	RUE ADVERSE	1845000	NAMUR	32-81-739140

```
85          CBA_KARATE_1     RUE ADVERSE   1855000            NAMUR
32-81-739141
86          CBA_KARATE_2     RUE ADVERSE   1865000            NAMUR
32-81-739142
87          CBA_KARATE_3     RUE ADVERSE   1875000            NAMUR
32-81-739143
88          CBA_KARATE_4     RUE ADVERSE   1885000            NAMUR
32-81-739144
89          CBA_KARATE_5     RUE ADVERSE   1895000            NAMUR
32-81-739145
90          CBA_KARATE_6     RUE ADVERSE   1905000            NAMUR
32-81-739146\
" > cbao.asc

echo "\
 116-JUN-8900                        6 - 4 1 1
 215-JUL-8900                        3 - 7 1 2
 312-AUG-8900                        8 - 2 1 3
 416-SEP-8901                            1 4
 516-OCT-8910                        5 - 5 1 5
 614-NOV-8911                            1 6
 718-JUN-8900                        1 - 9 2 7
 816-JUL-8900                        5 - 5 2 8
 915-AUG-8900                        6 - 4 2 9
1016-SEP-8901                            210
1116-OCT-8910                        8 - 2 211
1214-NOV-8911                            212
1318-JUN-8900                        8 - 2 385
1416-JUL-8900                        5 - 5 386
1514-AUG-8900                        4 - 6 387
1618-SEP-8901                            388
1712-OCT-8910                        3 - 7 389
1816-NOV-8911                            390
1914-JUN-8900                        6 - 6 413
2012-JUL-8900                        5 - 7 414
2115-AUG-8900                       10 - 2 415
2214-SEP-8901                            416
2318-OCT-8910                        6 - 6 417
2416-NOV-8911                            418
2512-JUN-8900                      2 - 10 519
2614-JUL-8900                        8 - 4 520
2718-AUG-8900                        9 - 3 521
2812-SEP-8901                            522
2914-OCT-8910                       10 - 2 523
3016-NOV-8911                            524
3115-JUN-8900                        2 - 2 625
3218-JUL-8900                        3 - 1 626
3312-AUG-8900                        4 - 0 627
3414-SEP-8901                            628
3516-OCT-8910                        1 - 3 629
3618-NOV-8911                            630
3712-JUN-8900                        3 - 0 731
3814-JUL-8900                        1 - 1 732
3916-AUG-8900                        0 - 2 733
4018-SEP-8901                            734
4112-OCT-8910                        2 - 2 735
4214-NOV-8911                            736
4316-JUN-8900     14/16 - 16/14 - 11/15 837
```

```
4418-JUL-8900     15/10 - 12/15 - 15/08 838
4512-AUG-8900     15/11 - 14/16 - 15/10 839
4614-SEP-8901                           840
4715-OCT-8910           15/04 - 15/10 841
4816-NOV-8911                           842
4912-JUN-8900     10/12 - 12/08 - 12/10 943
5018-JUL-8900           12/06 - 12/05 944
5114-AUG-8900           10/12 - 08/12 945
5216-SEP-8901                           946
5318-OCT-8910           08/12 - 10/12 947
5412-NOV-8911                           948
5514-JUN-8900           12/15 - 10/151049
5616-JUL-8900     15/09 - 10/15 - 15/111050
5715-AUG-8900           15/12 - 15/101051
5818-SEP-8901                          1052
5916-OCT-8910           10/15 - 09/151053
6012-NOV-8911                          1054
6115-JUN-8900                 6 - 61155
6214-JUL-8900                 4 - 81156
6316-AUG-8900                 7 - 51157
6412-SEP-8901                          1158
6514-OCT-8910                 6 - 61159
6614-NOV-8911                          1160
6716-JUN-8900           15/12 - 15/101261
6818-JUL-8900     15/11 - 10/15 - 09/151262
6916-AUG-8900     08/15 - 15/10 - 15/111263
7012-SEP-8901                          1264
7118-OCT-8910           15/10 - 15/121265
7216-NOV-8911                          1266
7314-JUN-8900                 3 - 21367
7418-JUL-8900                 1 - 41368
7515-AUG-8900                 4 - 11369
7612-SEP-8901                          1370
7716-OCT-8910                 2 - 31371
7814-NOV-8911                          1372
7916-JUN-8900             540 - 4311473
8015-JUL-8900             620 - 5441474
8114-AUG-8900             452 - 5511475
8212-SEP-8901                          1476
8318-OCT-8910             557 - 5031477
8416-NOV-8911                          1478
8518-JUN-8900             564 - 5011579
8614-JUL-8900             504 - 5561580
8716-AUG-8900             432 - 5081581
8814-SEP-8901                          1582
8915-OCT-8910             589 - 4981583
9018-NOV-8911                         1584\
" > rcto.asc

echo "\
J
H
G
B
P\
" > bidconso.asc

echo "\
```

```
A
E
I
O
U\
" > bidvoyo.asc

echo "\
R
T
P
N
M\
" > consrueo.asc

echo "\
1
3
5
7
9\
" > bidnum1o.asc

echo "\
0
2
4
6
8\
" > bidnum2o.asc

echo "\
1
2\
" > bidcot1o.asc


echo "\
0
1
2
3
4
5
6
7
8
9\
" > bidcot2o.asc

echo ""
echo "Chargement des tables de bases ..."
ctbo.ex

echo ""
echo "Creation de 15.626 noms ..."
echo "Temps requis pour la creation de 15.626 noms" >> timeo
echo "****************************************" >> timeo
```

```
time 2>>timeo sqlplus -silent system/manager @unloadmbn.sql

echo ""
echo "Limitation a 15.000 noms ..."
echo "Temps requis pour la limitation a 15.000 noms" >> timeo
echo "****************************************" >> timeo
time 2>>timeo awk 'NR > 625 {print "        "$1}' <
bidname1.lst > bidname2.lst

echo ""
echo "Quelques manipulations de fichiers ..."
awk '{print NR}' < bidname2.lst > bidname3.lst

echo ""
echo "Mise en forme des numeros de membre ..."
formenr.ex

echo ""
echo "Quelques manipulations de fichiers ..."
paste -d" " bidname3.lst bidname2.lst > bidname4.lst
awk 'NR <= 15000 {print "    "$1}' < bidname1.lst >
bidname5.lst
paste -d" " bidname4.lst bidname5.lst > bidname6.lst

echo ""
echo "Creation de 15.626 noms de rues..."
echo "Temps requis pour la creation de 15.626 noms de rues" >>
timeo
echo "**************************************************" >>
timeo
time 2>>timeo sqlplus -silent system/manager @unloadrue.sql

echo ""
echo "Limitation a 15.000 noms ..."
echo "Temps requis pour la limitation a 15.000 noms de rues"
>> timeo
echo "**************************************************"
>> timeo
time 2>>timeo awk 'NR > 625 {print "RUE "$1}' < bidrues1.lst >
bidrues2.lst

echo ""
echo "Quelques manipulations de fichiers ..."
paste -d" " bidname6.lst bidrues2.lst > bidname7.lst
awk '{print $0"  405000    NAMUR32-81"}' < bidname7.lst >
bidname8.lst

echo ""
echo "Creation de 15.626 numeros de telephone ..."
echo "Temps requis pour la creation de 15.626 numeros de
telephone" >> timeo
echo
"************************************************************"
>> timeo
time 2>>timeo sqlplus -silent system/manager @unloadnum.sql

echo ""
echo "Limitation a 15.000 numeros ..."
```

```
echo "Temps requis pour la limitation a 15.000 numeros" >>
timeo
echo "**********************************************" >>
timeo
time 2>>timeo awk 'NR > 625 {print $1}' < bidteln1.lst >
bidteln2.lst

echo ""
echo "Quelques manipulations de fichiers ..."
paste -d"-" bidname8.lst bidteln2.lst > bidname9.lst
awk '{print $0"01-JAN-601"}' < bidname9.lst > bidname10.lst

echo ""
echo "Creation de 2.000 cotisations ..."
echo "Temps requis pour la creation de 2.000 cotisations" >>
timeo
echo "**********************************************" >>
timeo
time 2>>timeo sqlplus -silent system/manager @unloadcot.sql

echo ""
echo "Quelques manipulations de fichiers pour atteindre 30.000
cotisations"
cp bidlic1o.lst bidlic2o.lst
cat bidlic1o.lst bidlic2o.lst > bidlic3o.lst
rm bidlic2o.lst
cp bidlic3o.lst bidlic4o.lst
cat bidlic3o.lst bidlic4o.lst > bidlic5o.lst
rm bidlic4o.lst
cp bidlic5o.lst bidlic6o.lst
cat bidlic5o.lst bidlic6o.lst > bidlic7o.lst
rm bidlic6o.lst
cat bidlic7o.lst bidlic5o.lst > bid1o.lst
rm bidlic5o.lst
rm bidlic7o.lst
cat bid1o.lst bidlic3o.lst > bid2o.lst
rm bidlic3o.lst
rm bid1o.lst
cat bid2o.lst bidlic1o.lst > bidlic8o.lst
rm bidlic1o.lst
rm bid2o.lst

echo ""
echo "Quelques manipulations supplementaires ..."
awk '{print $1}' < bidlic8o.lst > bidlic9o.lst
awk '{print NR}' < bidlic9o.lst > bidlic10.lst

echo ""
echo "Mise en forme des numeros de licence ..."
formeli.ex
paste -d" " bidlic10.lst bidlic9o.lst > bidlic11.lst

echo ""
echo "Creation des 30.000 numeros de clubs pour les licences
..."
echo "Temps requis pour la creation des 30.000 numeros de
clubs" >> timeo
```

```
echo
"*************************************************" >>
timeo
time 2>>timeo crnumcbo.ex

echo ""
echo "Creation des 30.000 numeros de membres pour les licences
..."
echo "Temps requis pour la creation des 30.000 numeros de
membres" >> timeo
echo
"*************************************************"
>> timeo
time 2>>timeo crnummbo.ex

echo ""
echo "Quelques manipulations de fichiers ..."
paste -d" " bidlic11.lst numcb.lst > bidlic12.lst
paste -d" " bidlic12.lst nummb.lst > bidlic13.lst

echo ""
echo "Creation d'un cluster ..."
echo "temps requis pour la creation d'un cluster" >> timeo
echo "*************************************" >> timeo
time 2>>timeo sqlplus -silent system/manager @ccluml.sql >
bidon2


echo ""
echo "Chargement des tables MEMBRE et LICENCE ..."
ctbocp.ex

echo ""
echo "Creation des uniques index ..."
echo "Temps requis pour la creation des index uniques" >>
timeo
echo "*********************************************" >>
timeo
time 2>>timeo sqlplus -silent system/manager @crunind.sql >
bidon

echo ""
echo "Destruction des tables et des fichiers intermediaires
..."
echo "Temps requis pour la destruction des tables
intermediaires ..." >> timeo
echo
"*************************************************
*" >> timeo
time 2>>timeo sqlplus -silent system/manager @deltabint.sql >
bidon
rm *.lst
rm *.asc
rm bidon
rm errlog

exit
```

```
#
# INFORMIX    -    FICHIER DE GENERATION DE LA BASE DE DONNEES
#
# INFORMIX is a registred trademark of INFORMIX SOFTWARE INC.
#


clear

echo ""
echo "Ce script va creer la Database -sport-, ses tables, et"
echo "ses records."
echo ""
echo "*****************************************************"
echo ""
echo "ENTER quand vous voulez ..."
read ATTENTE

clear

echo ""
echo "Creation de la Database et des tables ..."
echo "Temps requis pour la creation de la Database et des
tables ..." > ftime
echo
"*************************************************************
*">> ftime
time 2>>ftime cdbs.4ge

echo "\
1¦CSP_MULTI_SPORTS¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310555¦\
" > csp.asc

echo "\
1¦CB_JUDO¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310556¦1¦
2¦CB_AIKIDO¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310557¦1¦
3¦CB_KARATE¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310558¦1¦
4¦CB_TIR¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310559¦1¦
5¦CB_TIR_ARC¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310560¦1¦
6¦CB_ESCALADE¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310561¦1¦
7¦CB_FOOTBALL¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310562¦1¦
8¦CB_VOLLEYBALL¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310563¦1¦
9¦CB_HANDBALL¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310564¦1¦
10¦CB_BADMINTON¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310565¦1¦
11¦CB_NATATION¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310566¦1¦
12¦CB_SQUASH¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310567¦1¦
13¦CB_TENNIS¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310568¦1¦
14¦CB_GYMNASTIQUE¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310569¦1¦
15¦CB_ATHLETISME¦RUE VANDAMME, 40¦5000¦NAMUR¦32-81-3310570¦1¦\
" > cb.asc

echo "\
1¦CBA_JUDO_1¦RUE ADVERSE, 101¦5000¦NAMUR¦32-81-739056¦
2¦CBA_JUDO_2¦RUE ADVERSE, 102¦5000¦NAMUR¦32-81-739057¦
3¦CBA_JUDO_3¦RUE ADVERSE, 103¦5000¦NAMUR¦32-81-739058¦
4¦CBA_JUDO_4¦RUE ADVERSE, 104¦5000¦NAMUR¦32-81-739059¦
5¦CBA_JUDO_5¦RUE ADVERSE, 105¦5000¦NAMUR¦32-81-739060¦
6¦CBA_JUDO_6¦RUE ADVERSE, 106¦5000¦NAMUR¦32-81-739061¦
```

7¦CBA_AIKIDO_1¦RUE ADVERSE, 107¦5000¦NAMUR¦32-81-739062¦
8¦CBA_AIKIDO_2¦RUE ADVERSE, 108¦5000¦NAMUR¦32-81-739063¦
9¦CBA_AIKIDO_3¦RUE ADVERSE, 109¦5000¦NAMUR¦32-81-739064¦
10¦CBA_AIKIDO_4¦RUE ADVERSE, 110¦5000¦NAMUR¦32-81-739065¦
11¦CBA_AIKIDO_5¦RUE ADVERSE, 111¦5000¦NAMUR¦32-81-739066¦
12¦CBA_AIKIDO_6¦RUE ADVERSE, 112¦5000¦NAMUR¦32-81-739067¦
13¦CBA_TIR_1¦RUE ADVERSE, 113¦5000¦NAMUR¦32-81-739068¦
14¦CBA_TIR_2¦RUE ADVERSE, 114¦5000¦NAMUR¦32-81-739069¦
15¦CBA_TIR_3¦RUE ADVERSE, 115¦5000¦NAMUR¦32-81-739070¦
16¦CBA_TIR_4¦RUE ADVERSE, 116¦5000¦NAMUR¦32-81-739071¦
17¦CBA_TIR_5¦RUE ADVERSE, 117¦5000¦NAMUR¦32-81-739072¦
18¦CBA_TIR_6¦RUE ADVERSE, 118¦5000¦NAMUR¦32-81-739073¦
19¦CBA_TIR_ARC_1¦RUE ADVERSE, 119¦5000¦NAMUR¦32-81-739074¦
20¦CBA_TIR_ARC_2¦RUE ADVERSE, 120¦5000¦NAMUR¦32-81-739075¦
21¦CBA_TIR_ARC_3¦RUE ADVERSE, 121¦5000¦NAMUR¦32-81-739076¦
22¦CBA_TIR_ARC_4¦RUE ADVERSE, 122¦5000¦NAMUR¦32-81-739077¦
23¦CBA_TIR_ARC_5¦RUE ADVERSE, 123¦5000¦NAMUR¦32-81-739078¦
24¦CBA_TIR_ARC_6¦RUE ADVERSE, 124¦5000¦NAMUR¦32-81-739079¦
25¦CBA_ESCALADE_1¦RUE ADVERSE, 125¦5000¦NAMUR¦32-81-739080¦
26¦CBA_ESCALADE_2¦RUE ADVERSE, 126¦5000¦NAMUR¦32-81-739081¦
27¦CBA_ESCALADE_3¦RUE ADVERSE, 127¦5000¦NAMUR¦32-81-739082¦
28¦CBA_ESCALADE_4¦RUE ADVERSE, 128¦5000¦NAMUR¦32-81-739083¦
29¦CBA_ESCALADE_5¦RUE ADVERSE, 129¦5000¦NAMUR¦32-81-739084¦
30¦CBA_ESCALADE_6¦RUE ADVERSE, 130¦5000¦NAMUR¦32-81-739085¦
31¦CBA_FOOTBALL_1¦RUE ADVERSE, 131¦5000¦NAMUR¦32-81-739086¦
32¦CBA_FOOTBALL_2¦RUE ADVERSE, 132¦5000¦NAMUR¦32-81-739087¦
33¦CBA_FOOTBALL_3¦RUE ADVERSE, 133¦5000¦NAMUR¦32-81-739088¦
34¦CBA_FOOTBALL_4¦RUE ADVERSE, 134¦5000¦NAMUR¦32-81-739089¦
35¦CBA_FOOTBALL_5¦RUE ADVERSE, 135¦5000¦NAMUR¦32-81-739090¦
36¦CBA_FOOTBALL_6¦RUE ADVERSE, 136¦5000¦NAMUR¦32-81-739091¦
37¦CBA_VOLLEYBALL_1¦RUE ADVERSE, 137¦5000¦NAMUR¦32-81-739092¦
38¦CBA_VOLLEYBALL_2¦RUE ADVERSE, 138¦5000¦NAMUR¦32-81-739093¦
39¦CBA_VOLLEYBALL_3¦RUE ADVERSE, 139¦5000¦NAMUR¦32-81-739094¦
40¦CBA_VOLLEYBALL_4¦RUE ADVERSE, 140¦5000¦NAMUR¦32-81-739095¦
41¦CBA_VOLLEYBALL_5¦RUE ADVERSE, 141¦5000¦NAMUR¦32-81-739096¦
42¦CBA_VOLLEYBALL_6¦RUE ADVERSE, 142¦5000¦NAMUR¦32-81-739097¦
43¦CBA_HANDBALL_1¦RUE ADVERSE, 143¦5000¦NAMUR¦32-81-739098¦
44¦CBA_HANDBALL_2¦RUE ADVERSE, 144¦5000¦NAMUR¦32-81-739099¦
45¦CBA_HANDBALL_3¦RUE ADVERSE, 145¦5000¦NAMUR¦32-81-739100¦
46¦CBA_HANDBALL_4¦RUE ADVERSE, 146¦5000¦NAMUR¦32-81-739101¦
47¦CBA_HANDBALL_5¦RUE ADVERSE, 147¦5000¦NAMUR¦32-81-739102¦
48¦CBA_HANDBALL_6¦RUE ADVERSE, 148¦5000¦NAMUR¦32-81-739103¦
49¦CBA_BADMINTON_1¦RUE ADVERSE, 149¦5000¦NAMUR¦32-81-739104¦
50¦CBA_BADMINTON_2¦RUE ADVERSE, 150¦5000¦NAMUR¦32-81-739105¦
51¦CBA_BADMINTON_3¦RUE ADVERSE, 151¦5000¦NAMUR¦32-81-739106¦
52¦CBA_BADMINTON_4¦RUE ADVERSE, 152¦5000¦NAMUR¦32-81-739107¦
53¦CBA_BADMINTON_5¦RUE ADVERSE, 153¦5000¦NAMUR¦32-81-739108¦
54¦CBA_BADMINTON_6¦RUE ADVERSE, 154¦5000¦NAMUR¦32-81-739109¦
55¦CBA_NATATION_1¦RUE ADVERSE, 155¦5000¦NAMUR¦32-81-739110¦
56¦CBA_NATATION_2¦RUE ADVERSE, 156¦5000¦NAMUR¦32-81-739112¦
57¦CBA_NATATION_3¦RUE ADVERSE, 157¦5000¦NAMUR¦32-81-739113¦
58¦CBA_NATATION_4¦RUE ADVERSE, 158¦5000¦NAMUR¦32-81-739114¦
59¦CBA_NATATION_5¦RUE ADVERSE, 159¦5000¦NAMUR¦32-81-739115¦
60¦CBA_NATATION_6¦RUE ADVERSE, 160¦5000¦NAMUR¦32-81-739116¦
61¦CBA_SQUASH_1¦RUE ADVERSE, 161¦5000¦NAMUR¦32-81-739117¦
62¦CBA_SQUASH_2¦RUE ADVERSE, 162¦5000¦NAMUR¦32-81-739118¦
63¦CBA_SQUASH_3¦RUE ADVERSE, 163¦5000¦NAMUR¦32-81-739119¦
64¦CBA_SQUASH_4¦RUE ADVERSE, 164¦5000¦NAMUR¦32-81-739120¦

```
65¦CBA_SQUASH_5¦RUE ADVERSE, 165¦5000¦NAMUR¦32-81-739121¦
66¦CBA_SQUASH_6¦RUE ADVERSE, 166¦5000¦NAMUR¦32-81-739122¦
67¦CBA_TENNIS_1¦RUE ADVERSE, 167¦5000¦NAMUR¦32-81-739123¦
68¦CBA_TENNIS_2¦RUE ADVERSE, 168¦5000¦NAMUR¦32-81-739124¦
69¦CBA_TENNIS_3¦RUE ADVERSE, 169¦5000¦NAMUR¦32-81-739125¦
70¦CBA_TENNIS_4¦RUE ADVERSE, 170¦5000¦NAMUR¦32-81-739126¦
71¦CBA_TENNIS_5¦RUE ADVERSE, 171¦5000¦NAMUR¦32-81-739127¦
72¦CBA_TENNIS_6¦RUE ADVERSE, 172¦5000¦NAMUR¦32-81-739128¦
73¦CBA_GYMNASTIQUE_1¦RUE ADVERSE, 173¦5000¦NAMUR¦32-81-739129¦
74¦CBA_GYMNASTIQUE_2¦RUE ADVERSE, 174¦5000¦NAMUR¦32-81-739130¦
75¦CBA_GYMNASTIQUE_3¦RUE ADVERSE, 175¦5000¦NAMUR¦32-81-739131¦
76¦CBA_GYMNASTIQUE_4¦RUE ADVERSE, 176¦5000¦NAMUR¦32-81-739132¦
77¦CBA_GYMNASTIQUE_5¦RUE ADVERSE, 177¦5000¦NAMUR¦32-81-739133¦
78¦CBA_GYMNASTIQUE_6¦RUE ADVERSE, 178¦5000¦NAMUR¦32-81-739134¦
79¦CBA_ATHLETISME_1¦RUE ADVERSE, 179¦5000¦NAMUR¦32-81-739135¦
80¦CBA_ATHLETISME_2¦RUE ADVERSE, 180¦5000¦NAMUR¦32-81-739136¦
81¦CBA_ATHLETISME_3¦RUE ADVERSE, 181¦5000¦NAMUR¦32-81-739137¦
82¦CBA_ATHLETISME_4¦RUE ADVERSE, 182¦5000¦NAMUR¦32-81-739138¦
83¦CBA_ATHLETISME_5¦RUE ADVERSE, 183¦5000¦NAMUR¦32-81-739139¦
84¦CBA_ATHLETISME_6¦RUE ADVERSE, 184¦5000¦NAMUR¦32-81-739140¦
85¦CBA_KARATE_1¦RUE ADVERSE, 185¦5000¦NAMUR¦32-81-739141¦
86¦CBA_KARATE_2¦RUE ADVERSE, 186¦5000¦NAMUR¦32-81-739142¦
87¦CBA_KARATE_3¦RUE ADVERSE, 187¦5000¦NAMUR¦32-81-739143¦
88¦CBA_KARATE_4¦RUE ADVERSE, 188¦5000¦NAMUR¦32-81-739144¦
89¦CBA_KARATE_5¦RUE ADVERSE, 189¦5000¦NAMUR¦32-81-739145¦
90¦CBA_KARATE_6¦RUE ADVERSE, 190¦5000¦NAMUR¦32-81-739146¦\
" > cba.asc

echo "\
1¦06/16/89¦0¦0¦6 - 4¦1¦1¦
2¦07/15/89¦0¦0¦3 - 7¦1¦2¦
3¦08/12/89¦0¦0¦8 - 2¦1¦3¦
4¦09/16/89¦0¦1¦¦1¦4¦
5¦10/16/89¦1¦0¦5 - 5¦1¦5¦
6¦11/14/89¦1¦1¦¦1¦6¦
7¦06/18/89¦0¦0¦1 - 9¦2¦7¦
8¦07/16/89¦0¦0¦5 - 5¦2¦8¦
9¦08/15/89¦0¦0¦6 - 4¦2¦9¦
10¦09/16/89¦0¦1¦¦2¦10¦
11¦10/16/89¦1¦0¦8 - 2¦2¦11¦
12¦11/14/89¦1¦1¦¦2¦12¦
13¦06/18/89¦0¦0¦8 - 2¦3¦85¦
14¦07/16/89¦0¦0¦5 - 5¦3¦86¦
15¦08/14/89¦0¦0¦4 - 6¦3¦87¦
16¦09/18/89¦0¦1¦¦3¦88¦
17¦10/12/89¦1¦0¦3 - 7¦3¦89¦
18¦11/16/89¦1¦1¦¦3¦90¦
19¦06/14/89¦0¦0¦6 - 6¦4¦13¦
20¦07/12/89¦0¦0¦5 - 7¦4¦14¦
21¦08/15/89¦0¦0¦10 - 2¦4¦15¦
22¦09/14/89¦0¦1¦¦4¦16¦
23¦10/18/89¦1¦0¦6 - 6¦4¦17¦
24¦11/16/89¦1¦1¦¦4¦18¦
25¦06/12/89¦0¦0¦2 - 10¦5¦19¦
26¦07/14/89¦0¦0¦8 - 4¦5¦20¦
27¦08/18/89¦0¦0¦9 - 3¦5¦21¦
28¦09/12/89¦0¦1¦¦5¦22¦
29¦10/14/89¦1¦0¦10 - 2¦5¦23¦
```

30¦11/16/89¦1¦1¦¦5¦24¦
31¦06/15/89¦0¦0¦2 - 2¦6¦25¦
32¦07/18/89¦0¦0¦3 - 1¦6¦26¦
33¦08/12/89¦0¦0¦4 - 0¦6¦27¦
34¦09/14/89¦0¦1¦¦6¦28¦
35¦10/16/89¦1¦0¦1 - 3¦6¦29¦
36¦11/18/89¦1¦1¦¦6¦30¦
37¦06/12/89¦0¦0¦3 - 0¦7¦31¦
38¦07/14/89¦0¦0¦1 - 1¦7¦32¦
39¦08/16/89¦0¦0¦0 - 2¦7¦33¦
40¦09/18/89¦0¦1¦¦7¦34¦
41¦10/12/89¦1¦0¦2 - 2¦7¦35¦
42¦11/14/89¦1¦1¦¦7¦36¦
43¦06/16/89¦0¦0¦14/16 - 16/14 - 11/15¦8¦37¦
44¦07/18/89¦0¦0¦15/10 - 12/15 - 15/08¦8¦38¦
45¦08/12/89¦0¦0¦15/11 - 14/16 - 15/10¦8¦39¦
46¦09/14/89¦0¦1¦¦8¦40¦
47¦10/15/89¦1¦0¦15/04 - 15/10¦8¦41¦
48¦11/16/89¦1¦1¦¦8¦42¦
49¦06/12/89¦0¦0¦10/12 - 12/08 - 12/10¦9¦43¦
50¦07/18/89¦0¦0¦12/06 - 12/05¦9¦44¦
51¦08/14/89¦0¦0¦10/12 - 08/12¦9¦45¦
52¦09/16/89¦0¦1¦¦9¦46¦
53¦10/18/89¦1¦0¦08/12 - 10/12¦9¦47¦
54¦11/12/89¦1¦1¦¦9¦48¦
55¦06/14/89¦0¦0¦12/15 - 10/15¦10¦49¦
56¦07/16/89¦0¦0¦15/09 - 10/15 - 15/11¦10¦50¦
57¦08/15/89¦0¦0¦15/12 - 15/10¦10¦51¦
58¦09/18/89¦0¦1¦¦10¦52¦
59¦10/16/89¦1¦0¦10/15 - 09/15¦10¦53¦
60¦11/12/89¦1¦1¦¦10¦54¦
61¦06/15/89¦0¦0¦6 - 6¦11¦55¦
62¦07/14/89¦0¦0¦4 - 8¦11¦56¦
63¦08/16/89¦0¦0¦7 - 5¦11¦57¦
64¦09/12/89¦0¦1¦¦11¦58¦
65¦10/14/89¦1¦0¦6 - 6¦11¦59¦
66¦11/14/89¦1¦1¦¦11¦60¦
67¦06/16/89¦0¦0¦15/12 - 15/10¦12¦61¦
68¦07/18/89¦0¦0¦15/11 - 10/15 - 09/15¦12¦62¦
69¦08/16/89¦0¦0¦08/15 - 15/10 - 15/11¦12¦63¦
70¦09/12/89¦0¦1¦¦12¦64¦
71¦10/18/89¦1¦0¦15/10 - 15/12¦12¦65¦
72¦11/16/89¦1¦1¦¦12¦66¦
73¦06/14/89¦0¦0¦3 - 2¦13¦67¦
74¦07/18/89¦0¦0¦1 - 4¦13¦68¦
75¦08/15/89¦0¦0¦4 - 1¦13¦69¦
76¦09/12/89¦0¦1¦¦13¦70¦
77¦10/16/89¦1¦0¦2 - 3¦13¦71¦
78¦11/14/89¦1¦1¦¦13¦72¦
79¦06/16/89¦0¦0¦540 - 431¦14¦73¦
80¦07/15/89¦0¦0¦620 - 544¦14¦74¦
81¦08/14/89¦0¦0¦452 - 551¦14¦75¦
82¦09/12/89¦0¦1¦¦14¦76¦
83¦10/18/89¦1¦0¦557 - 503¦14¦77¦
84¦11/16/89¦1¦1¦¦14¦78¦
85¦06/18/89¦0¦0¦564 - 501¦15¦79¦
86¦07/14/89¦0¦0¦504 - 556¦15¦80¦
87¦08/16/89¦0¦0¦432 - 508¦15¦81¦

```
88¦09/14/89¦0¦1¦¦15¦82¦
89¦10/15/89¦1¦0¦589 - 498¦15¦83¦
90¦11/18/89¦1¦1¦¦15¦84¦\
" > rct.asc

echo "\
J¦
H¦
G¦
B¦
P¦\
" > bidcons.asc

echo "\
A¦
E¦
I¦
O¦
U¦\
" > bidvoy.asc

echo "\
R¦
T¦
P¦
N¦
M¦\
" > consrue.asc

echo "\
1¦
3¦
5¦
7¦
9¦\
" > bidnum1.asc

echo "\
0¦
2¦
4¦
6¦
8¦\
" > bidnum2.asc

echo "\
1¦
2¦\
" > bidcot1.asc

echo "\
0¦
1¦
2¦
3¦
4¦
5¦
6¦
```

```
7¦
8¦
9¦\
" > bidcot2.asc

echo ""
echo "Chargement des tables de base ..."
ctbi.ex

echo ""
echo "Creation de 15.626 noms ..."
echo "Temps requis pour la creation de 15.626 noms" >> ftime
echo "*****************************************" >> ftime
time 2>>ftime unloadmbn.4ge

echo ""
echo "Limitation a 15.000 noms ..."
echo "Temps requis pour la limitation a 15.000 noms" >> ftime
echo "*****************************************" >> ftime
time 2>>ftime awk 'NR > 625 {print $0}' < bidname.asc >
bidname2.asc

echo ""
echo "Quelques manipulations de fichiers ..."
awk '{print NR}' < bidname2.asc > bidname3.asc
paste -d"¦" bidname3.asc bidname2.asc > bidname4.asc
rm bidname2.asc
rm bidname3.asc
awk 'NR <= 15000 {print $0}' < bidname.asc > bidname5.asc
paste -d"¦" bidname4.asc bidname5.asc > bidname6.asc
rm bidname.asc
rm bidname4.asc
rm bidname5.asc

echo ""
echo "Creation de 15.626 noms de rue ..."
echo "Temps requis pour la creation de 15.626 noms de rue" >>
ftime
echo "*************************************************" >>
ftime
time 2>>ftime unloadrue.4ge

echo ""
echo "Limitation a 15.000 noms ..."
echo "Temps requis pour la limitation a 15.000 noms de rue" >>
ftime
echo "*************************************************" >>
ftime
time 2>>ftime awk 'NR > 625 {print $0}' < bidrue.asc >
bidrue2.asc
rm bidrue.asc

echo ""
echo "Quelques manipulations de fichiers ..."
awk '{print "RUE "$0}' < bidrue2.asc > bidrue3.asc
rm bidrue2.asc
paste -d"¦" bidname6.asc bidrue3.asc > bidname7.asc
rm bidname6.asc
```

```
rm bidrue3.asc
awk '{print $0", 40¦5000¦NAMUR¦32-81"}'< bidname7.asc >
bidname8.asc
rm bidname7.asc

echo ""
echo "Creation de 15.626 numeros de telephone ..."
echo "Temps requis pour la creation de 15.626 numeros de
telephone" >> ftime
echo
"*************************************************************"
>> ftime
time 2>>ftime unloadnum.4ge

echo ""
echo "Limitation a 15.000 numeros ..."
echo "Temps requis pour la limitation a 15.000 numeros" >>
ftime
echo "************************************************" >>
ftime
time 2>>ftime awk 'NR > 625 {print $0}' < bidtel.asc >
bidtel2.asc
rm bidtel.asc

echo ""
echo "Quelques manipulations de fichiers ..."
paste -d"-" bidname8.asc bidtel2.asc > bidname9.asc
rm bidname8.asc
rm bidtel2.asc
awk '{print $0"¦01/01/60¦1¦"}' < bidname9.asc > bidname10.asc
rm bidname9.asc
awk 'NR <= 15000 {print $0}' < bidname10.asc > bidname11.asc
rm bidname10.asc

echo ""
echo "Creation de 2.000 cotisations ..."
echo "Temps requis pour la creation de 2.000 cotisations" >>
ftime
echo "************************************************" >>
ftime
time 2>>ftime unloadcot.4ge

echo ""
echo "Quelques manipulations de fichiers pour atteindre 30.000
cotisations"
cp bidlic.asc bidlic1.asc
cat bidlic.asc bidlic1.asc > bidlic2.asc
rm bidlic.asc
cp bidlic2.asc bidlic3.asc
cat bidlic2.asc bidlic3.asc > bidlic4.asc
rm bidlic3.asc
cp bidlic4.asc bidlic5.asc
cat bidlic4.asc bidlic5.asc > bidlic6.asc
rm bidlic4.asc
cat bidlic6.asc bidlic5.asc > bidl1.asc
rm bidlic5.asc
rm bidlic6.asc
cat bidl1.asc bidlic2.asc > bidl2.asc
```

```
rm bidlic2.asc
rm bidl1.asc
cat bidl2.asc bidlic1.asc > bidlic7.asc
rm bidlic1.asc
rm bidl2.asc

echo ""
echo "Quelques manipulations supplementaires ..."
awk 'NF > 0 {print}' < bidlic7.asc > bidlic8.asc
rm bidlic7.asc
awk '{print NR}' < bidlic8.asc > bidlic9.asc
paste -d"¦" bidlic9.asc bidlic8.asc > bidlic10.asc
rm bidlic9.asc
rm bidlic8.asc

echo ""
echo "Creation des 30.000 num de clubs pour les licences ..."
echo "Temps requis pour la creation des 30.000 numeros de
clubs" >> ftime
echo
"**************************************************" >>
ftime
time 2>>ftime crnumcb.ex
echo ""
echo "Creation des 30.000 num de membres pour les licences
..."
echo "Temps requis pour la creation des 30.000 numeros de
membres" >> ftime
echo
"**************************************************"
>> ftime
time 2>>ftime crnummb.ex
echo ""
echo "Quelques manipulations de fichiers ..."
paste -d"¦" bidlic10.asc numcb.asc > bidlic11.asc
rm bidlic10.asc
rm numcb.asc
paste -d"¦" bidlic11.asc nummb.asc > bidlic12.asc
rm bidlic11.asc
rm nummb.asc
awk '{print $0"¦"}' < bidlic12.asc > bidlic13.asc
rm bidlic12.asc

echo ""
echo "Chargement des tables MEMBRE et LICENCE ..."
ctbicp.ex

echo ""
echo "Creation des index uniques ..."
echo "Temps requis pour la creation des index uniques" >>
ftime
echo "**************************************************" >>
ftime
echo "time 2>>ftime crunind.4ge"

echo ""
echo "Destruction des tables et des fichiers intermediaires
..."
```

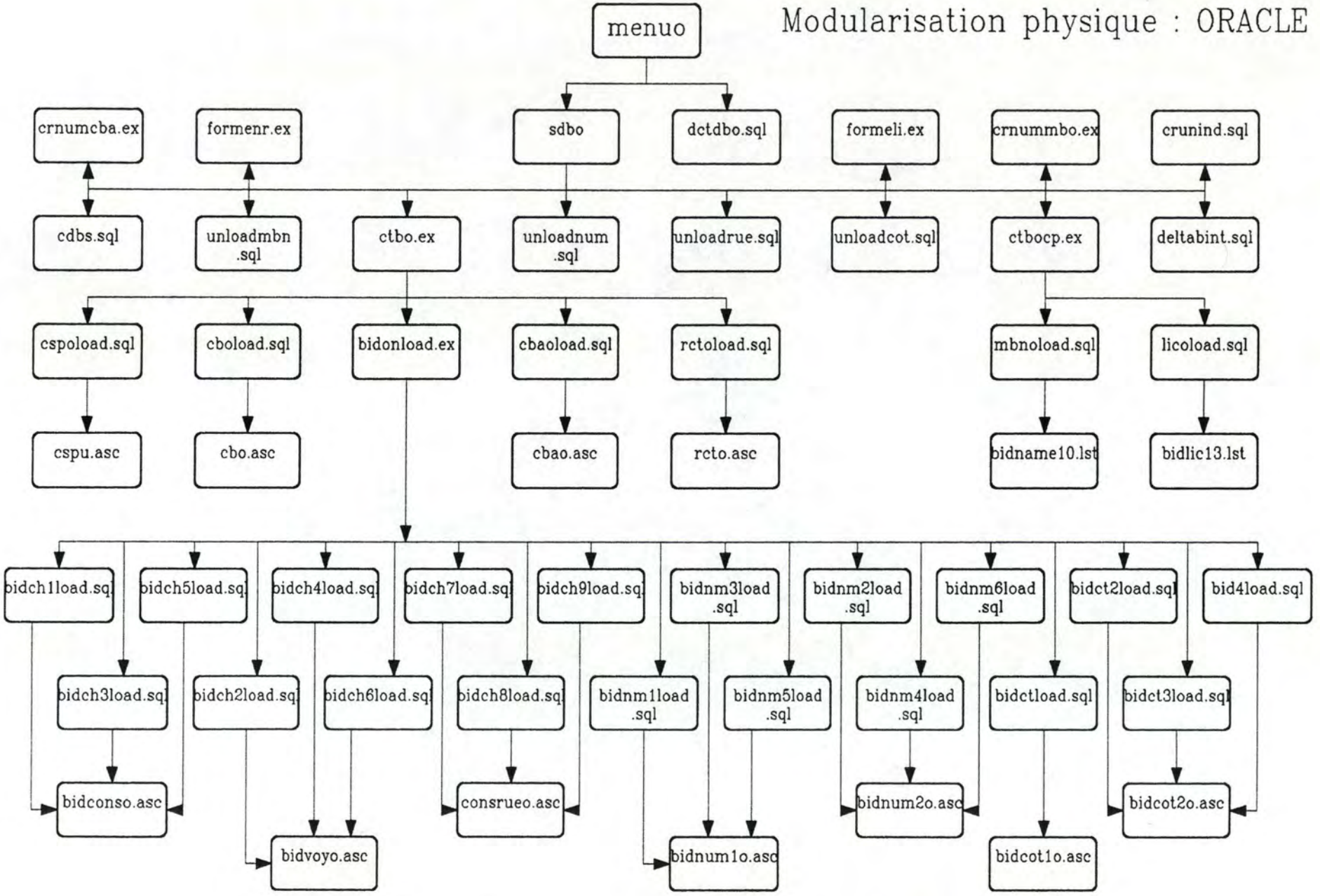
```
echo "Temps requis pour la destruction des tables
intermediaires" >> ftime
echo
"*************************************************************"
>> ftime
time 2>>ftime deltabint.4ge
rm *.asc

exit
```

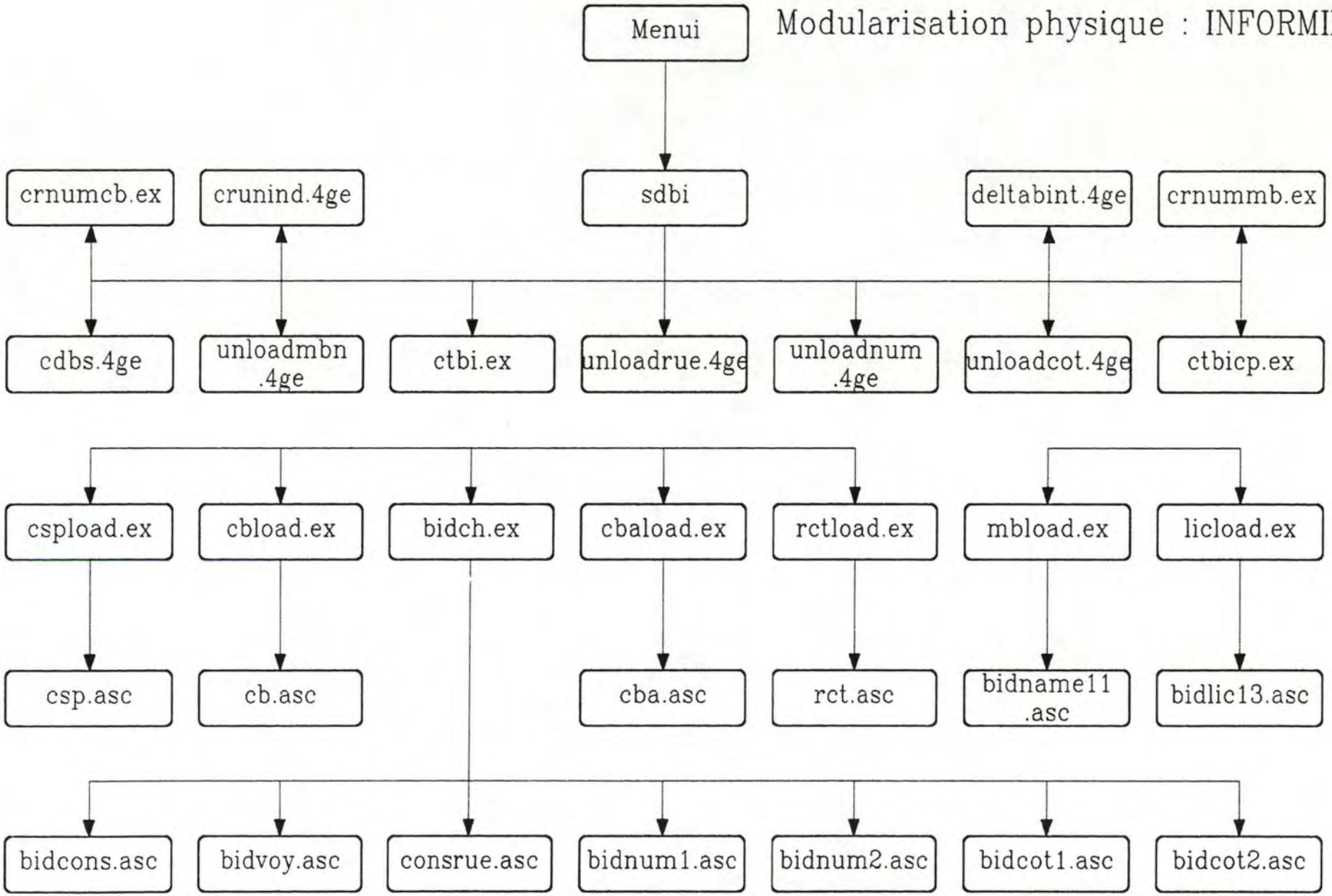
Annexe 2

Hiérarchisation physique des modules pour la création automatique de la base de données

Modularisation physique : ORACLE



Modularisation physique : INFORMIX



Annexe 3

Codes associés aux requêtes utilisées pour les mesures de performances

```
REM
REM ORACLE   -   FICHIER DE COMMANDES POUR INSERER
REM                 DE NOUVEAUX ENREGISTREMENTS
REM
REM      ORACLE is a registred trademark of ORACLE CORPORATION

INSERT INTO RESERVATION (NRS, Nsalle, Nterrain, Date_RS,
Heure, NM)
   VALUES (0, 1, 1, '22-JAN-90', 14.00, 1);

EXIT
```

```
REM
REM ORACLE   -      FICHIER DE COMMANDES POUR SUPPRIMER
REM                          DES ENREGISTREMENTS
REM
REM ORACLE is a registred trademark of ORACLE CORPORATION
REM

DELETE FROM RESERVATION;

EXIT
```

```
REM
REM ORACLE DATABASE   -   FICHIER DE COMMANDES POUR LA
REM                       MISE A JOUR D'UN ENREGISTREMENT
REM
REM ORACLE is a registred trademark of ORACLE CORPORATION

UPDATE LICENCE
   SET COTISATION = COTISATION + 1
   WHERE LICENCE.NL = MOD (TO_NUMBER (TO_CHAR
(SYSDATE,'SSSSS')),20000);

EXIT
```

```
REM
REM ORACLE   -   FICHIER DE COMMANDES POUR MESURER
REM                DES TEMPS DE SELECTION SIMPLE
REM
REM ORACLE is a registred trademark of ORACLE CORPORATION

SELECT MEMBRE.NM, MEMBRE.Nom, MEMBRE.Prenom, MEMBRE.Adresse,
       MEMBRE.Code_P, MEMBRE.Ville, MEMBRE.Tel,
       MEMBRE.Date_de_N, MEMBRE.NCS
FROM  MEMBRE, LICENCE
```

```
WHERE LICENCE.NC = 1
      AND
      LICENCE.NM = MEMBRE.NM;

EXIT
```

```
REM
REM ORACLE   -   FICHIER DE COMMANDES POUR MESURER
REM               DES TEMPS DE SELECTION COMPLEXE
REM
REM ORACLE is a registred trademark of ORACLE CORPORATION

SELECT MEMBRE.NM, LICENCE.NC, MEMBRE.Nom, MEMBRE.Prenom,
       MEMBRE.Adresse, MEMBRE.Code_P, MEMBRE.Ville,
       LICENCE.Cotisation
FROM MEMBRE, LICENCE
WHERE (LICENCE.NC = 1 or LICENCE.NC = 2)
      AND
      (LICENCE.NM = MEMBRE.NM)
      AND
      (LICENCE.Cotisation < 1800)
         MEMBRE.Code_P, MEMBRE.Ville, LICENCE.Cotisation
ORDER BY LICENCE.NC, MEMBRE.Nom;

EXIT
```

```
#
# INFORMIX   -   FICHIER DE COMMANDES POUR MESURER
#          LE TEMPS NECESSAIRE A L'AJOUT ET LA SUPPRESION
#                      DE AJMAX ENREGISTREMENTS
#
# INFORMIX is a registred trademark of INFORMIX SOFTWARE INC.
#

DATABASE sport

GLOBALS

   DEFINE ajmax     integer
   DEFINE k_nrs     integer

END GLOBALS

MAIN

   DEFINE retcode       integer

#      ajmax vaut 500

   CALL starttime (23,"rasinf",7) RETURNING retcode

   if retcode <> 0
      THEN DISPLAY "ERROR : starttime call at line 120."
   END IF

   LET ajmax = 500

   CALL ajenr (ajmax)

   CALL stoptime (33,"rasinf",7) RETURNING retcode

   IF retcode <> 0
      THEN DISPLAY "ERROR : stoptime call at line 129."
   END IF

   CALL starttime (39,"rasinf",7) RETURNING retcode

   IF retcode <> 0
      THEN DISPLAY "ERROR : starttime call at line 135."
   END IF

   CALL supenr ()

   CALL stoptime (47,"rasinf",7) RETURNING retcode

   IF retcode <> 0
      THEN DISPLAY "ERROR : stoptime call at line 143."
   END IF

end MAIN

FUNCTION ajenr (ajmax)

   DEFINE i         integer,
```

```
        var_res RECORD LIKE RESERVATION.*

   LET k_nrs = 100 + ARG_VAL(1)

   FOR i = 1 to ajmax

      LET var_res.NRS = # SQLCA.SQLERRD[2]
                          k_nrs
      LET var_res.Nsalle = 1
      LET var_res.Nterrain = 1
      LET var_res.Date_RS = "01/22/90"
      LET var_res.Heure = 14.00
      LET var_res.NM = i

      INSERT INTO RESERVATION
         VALUES (var_res.*)

   END FOR

END FUNCTION

FUNCTION supenr ()

  DELETE FROM RESERVATION
    WHERE NRS = k_nrs
END FUNCTION
```

```
#
# INFORMIX    -    FICHIER DE COMMANDES POUR MESURER LE
#                    TEMPS NECESSAIRE A LA MISE A JOUR
#                         MAJMAX ENREGISTREMENTS
#
# INFORMIX is a registred trademark of INFORMIX SOFTWARE INC.
#

DATABASE sport

GLOBALS

    DEFINE retcode      integer,
           nmml         integer,
           nb_enrs      integer,
           majmax       integer

END GLOBALS

MAIN

#       majmax vaut 1000

   CALL starttime (24,"rmajinf",8) RETURNING retcode

   IF retcode <> 0
      THEN DISPLAY "ERROR : starttime call at line 78."
```

```
   END IF

   LET majmax = 1000

   CALL majenr (majmax)

   CALL stoptime (34,"rmajinf",8) RETURNING retcode

   IF retcode <> 0
      THEN DISPLAY "ERROR : stoptime call at line 88."
   END IF

END MAIN

FUNCTION majenr (majmax)

   DEFINE i        integer

   LET nb_enrs = 20000

   FOR i=1 TO majmax

      CALL detrand (nb_enrs,i) RETURNING retcode, nmml

      IF retcode <> 0
         THEN DISPLAY "ERROR : detrand call at line 40."
      END IF

      UPDATE LICENCE SET COTISATION = COTISATION + 1
         WHERE NL = nmml

   END FOR

END FUNCTION
```

```
#
# INFORMIX    -    FICHIER DE COMMANDES POUR MESURER
#                LE TEMPS NECESSAIRE A LA SELECTION DE
#                       SELMAX ENREGISTREMENTS
#
# INFORMIX is a registred trademark of INFORMIX SOFTWARE INC.
#

DATABASE sport

GLOBALS

   DEFINE  selmax     integer

END GLOBALS

MAIN

   DEFINE retcode      integer
```

```
#    selmax vaut 20000

    CALL starttime (23,"rselinf",8) RETURNING retcode

    IF retcode <> 0
       THEN DISPLAY "ERROR : starttime call at line 277."
    END IF

    LET selmax = 20000

    CALL selenr ()

    CALL stoptime (33,"rselinf",8) RETURNING retcode

    IF retcode <>  0
       THEN DISPLAY "ERROR : stoptime call at line 287."
    END IF

END MAIN

FUNCTION selenr ()

   DEFINE i        integer

  LET i = 0
  WHILE i <= selmax

      DECLARE j CURSOR FOR
         SELECT MEMBRE.NM, MEMBRE.Nom, MEMBRE.Prenom,
                MEMBRE.Adresse, MEMBRE.Code_P, MEMBRE.Ville,
                MEMBRE.Tel, MEMBRE.DATE_de_N, MEMBRE.NCS
         FROM MEMBRE, LICENCE
         WHERE LICENCE.NC = 1
               and
               LICENCE.NM = MEMBRE.NM

       LET i = i+1

   END WHILE

END FUNCTION
```

```
#
# INFORMIX    -    FICHIER DE COMMANDES POUR MESURER
#                LE TEMPS NECESSAIRE A LA SELECTION DE
#                       SELMAX ENREGISTREMENTS
#
# INFORMIX is a registred trademark of INFORMIX SOFTWARE INC.
#

DATABASE sport

GLOBALS
```

```
   DEFINE  selmax     integer

END GLOBALS

MAIN

   DEFINE retcode      integer

#    selmax vaut 20.000

   CALL starttime (23,"rsel2inf",9) RETURNING retcode

   IF retcode <> 0
      THEN DISPLAY "ERROR : starttime call at line 257."
   END IF

   LET selmax = 20000

   CALL selenr ()

   CALL stoptime (33,"rsel2inf",9) RETURNING retcode

   IF retcode <>  0
      THEN DISPLAY "ERROR : stoptime call at line 267."
   END IF

END MAIN

FUNCTION selenr ()

   DEFINE i        integer

   LET i = 0
   WHILE i <= selmax

      DECLARE j CURSOR FOR
         SELECT MEMBRE.NM, LICENCE.NC, MEMBRE.Nom,
                MEMBRE.Prenom, MEMBRE.Adresse, MEMBRE.Code_P,
                MEMBRE.Ville, LICENCE.Cotisation
         FROM MEMBRE, CLUB, LICENCE
         WHERE (LICENCE.NC = 1 or LICENCE.NC = 2)
               AND
               (LICENCE.NM = MEMBRE.NM)
               AND
               (LICENCE.Cotisation < 1800)
         ORDER BY LICENCE.NC, MEMBRE.Nom

      LET i = i+1

   END WHILE

END FUNCTION
```